

Computation and Visualisation of Phase Portraits for Model Checking SPDIs

Gordon Pace¹ and Gerardo Schneider²

¹ Dept. of Computer Science, University of Malta, Msida, Malta
gordon.pace@um.edu.mt

² Dept. of Informatics, University of Oslo — PO Box 1080 Blindern, N-0316 Oslo, Norway
gerardo@ifi.uio.no

1 Introduction and Background

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. Reachability analysis has been the principal research question in the verification of hybrid systems, even though it is a well-known result that most non-trivial subclasses of hybrid systems reachability and most verification problems are undecidable [1]. Nonetheless, various decidable subclasses have been identified, including polygonal hybrid systems (SPDIs) [2]. SPDIs can be used, for instance, in the analysis of approximations of non-linear differential equations in two-dimensions.

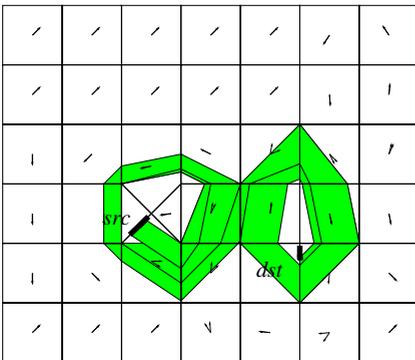


Fig. 1. Reachability on SPDI

Qualitative analysis of hybrid systems is an alternative but rather neglected research direction [3,4,5,6]. Typical qualitative questions include: “Are there ‘sink’ regions which one can never leave once they have been entered?”; “Which are the basins of attraction of such regions?”; “Are there regions in which every point in the region is reachable from any other point in the same region without leaving it?”. Answering such questions usually implies giving a collection of objects characterising such sets, which provide useful information about the qualitative

behaviour of the hybrid system. We call the set of all such objects the *phase portrait* of the system.

Defining and constructing phase portraits of hybrid systems has been directly addressed for SPDIs in [7,8]. In this paper we present a tool implementing the generation of phase portraits for SPDIs following the latter papers, and we show how these can be used to optimise the reachability analysis, in some cases even giving an immediate answer, as exposed in [9].

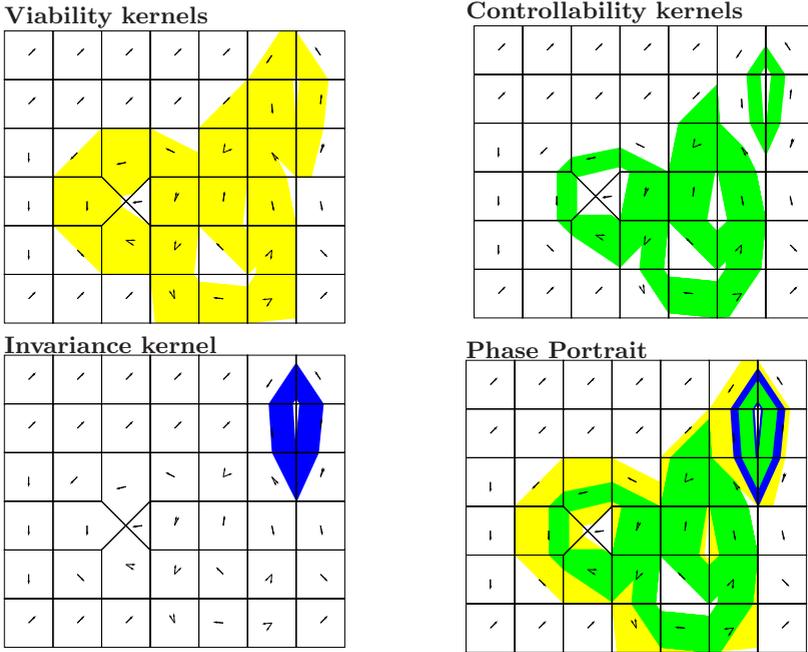


Fig. 2. Example generated by SPeeDI⁺

An *SPDI* (Fig. 1) consists of a finite partition \mathbb{P} of the plane (into convex polygonal areas), such that, each $P \in \mathbb{P}$ is associated to a pair of vectors \mathbf{a}_P and \mathbf{b}_P (shown as arrows in the polygons in the figure). The SPDI behaviour is defined by the differential inclusion $\dot{\mathbf{x}} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$ 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 [2] it has been proved that edge-to-edge and polygon-to-polygon reachability is decidable by exploiting the topological properties of the plane. The information gathered for computing reachability turns out to be useful for computing certain phase portrait objects of SPDIs. Given a cycle on a SPDI, we can speak about a number of kernels pertaining to that cycle [7,8]. The *viability* kernel is the largest set of points in the cycle which may loop forever within the cycle. The *controllability* kernel is the largest set of strongly connected points in the cycle. The *invariant kernel* is the largest set of points in a loop such that each point must keep rotating within the set forever.

Kernels are not only interesting as a mathematical curiosity but are crucial in model checking. The invariance kernel, for instance, has been used to prove termination in a breadth-first search algorithm for model checking SPDIs [10]. It is also of interest since it is much cheaper than reachability analysis, and one can use the kernels to abstract and reduce the size of SPDIs [9].

2 SPeeDI⁺

The tool-set SPeeDI [11] is a collection of utilities to manipulate and reason mechanically about SPDIs, implemented in Haskell including:

Visualisation aids: Graphical representations of SPDI, including simulation of trajectories and signatures within it.

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

Reachability analysis: SPeeDI allows the user to verify a system by checking reachability between restricted edges. It also enables the use of signatures (abstract paths through an SPDI), to enable exploration of feasible paths in an SPDI.

Trace generation: Whenever reachability analysis succeeds, SPeeDI generates an abstract signature as a witness. Since a signature embodies a collection of concrete paths through the SPDI, SPeeDI also provides tools to generate concrete paths from abstract signatures.

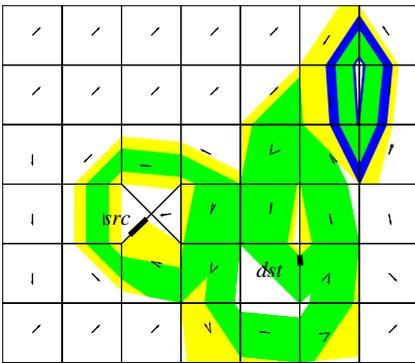


Fig. 3. Immediate answer using the kernels

given SPDI. Similarly, the top right figure depicts the controllability kernels and the bottom left show the unique invariance kernel. Note that some loops do not have a controllability kernel. The bottom right figure shows the complete phase portrait — all the kernels of the SPDI. The execution time for obtaining all the kernels in this example is instantaneous.

3 Applications and Discussion

SPeeDI⁺ implements the algorithms published in [7,8] based on the analysis of qualitative behaviours generated by a discrete dynamical system characterised by positive affine Poincaré maps. Currently there are no other tools specifically for SPDI analysis. Tools for generic hybrid systems, such as HyTech, are more generic, but are subsequently only semi-algorithms and less efficient than SPeeDI⁺. See [11] for a direct comparison.

We use the kernels computed with SPeeDI⁺ for optimising the reachability analysis, in some cases giving an immediate answer without exploring the state space. For example, the reachability question answered by the path given in Fig. 1 (and generated using the reachability analysis in SPeeDI) can be answered immediately

Despite the fact that lazy functional languages have a rather bad reputation regarding performance, the performance we obtained was more than adequate for our examples.

In this paper we present SPeeDI⁺, which extends our earlier tool SPeeDI, enabling the computation and analysis of three important phase portrait objects of an SPDI, namely viability, controllability and invariance kernels. Fig. 2 shows all the kernels for the SPDI depicted in Fig. 1. The top left figure shows the union of the viability kernels of ten (overlapping) loops present in the

just by observing the phase portrait picture (without performing reachability analysis) as can be seen in Fig. 3. Using a property of viability kernels, we know that there is a trajectory starting in the initial interval which will reach a point in the controllability kernel of the same cycle. Furthermore, by definition of the controllability kernel, the final interval (on another controllability kernel which intersects the first) is then reachable from the initial point.

As already noted, we can also use the kernels to abstract and reduce the state-space of a given SPDI. For example, when verifying reachability between two edges both lying within an invariance kernel, we can reduce the SPDI by discarding all regions outside the kernel, since by definition of the invariance kernel they can play no role in the reachability analysis, as it is not possible to leave the kernel. Similarly viability and controllability kernels can be used to decompose a reachability question to smaller ones by splitting the state space using the outline of these kernels as boundaries, and performing model checking on the smaller spaces independently. The theoretical results concerning state-space reduction and optimisation using kernels (and semi-separatrices) have been presented in [9], while the results showing how kernels can also be used to decompose reachability questions thus effectively giving a parallel algorithm for SPDI reachability has been presented in [12].

References

1. Henzinger, T., Kopke, P., Puri, A., Varaiya, P.: What's decidable about hybrid automata? In: STOC 1995, pp. 373–382. ACM Press, New York (1995)
2. Asarin, E., Schneider, G., Yovine, S.: On the decidability of the reachability problem for planar differential inclusions. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 89–104. Springer, Heidelberg (2001)
3. Aubin, J.P., Lygeros, J., Quincampoix, M., Sastry, S., Seube, N.: Viability and invariance kernels of impulse differential inclusions. In: Conference on Decision and Control. IEEE, vol. 40, pp. 340–345 (2001)
4. Aubin, J.P.: The substratum of impulse and hybrid control systems. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 105–118. Springer, Heidelberg (2001)
5. Deshpande, A., Varaiya, P.: Viable control of hybrid systems. In: Antsaklis, P.J., Kohn, W., Nerode, A., Sastry, S.S. (eds.) HS 1994. LNCS, vol. 999, pp. 128–147. Springer, Heidelberg (1995)
6. Kourjanski, M., Varaiya, P.: Stability of hybrid systems. In: Alur, R., Sontag, E.D., Henzinger, T.A. (eds.) HS 1995. LNCS, vol. 1066, pp. 413–423. Springer, Heidelberg (1996)
7. Asarin, E., Schneider, G., Yovine, S.: Towards computing phase portraits of polygonal differential inclusions. In: Tomlin, C.J., Greenstreet, M.R. (eds.) HSCC 2002. LNCS, vol. 2289, pp. 49–61. Springer, Heidelberg (2002)
8. Schneider, G.: Computing invariance kernels of polygonal hybrid systems. *Nordic Journal of Computing* 11, 194–210 (2004)
9. Pace, G., Schneider, G.: Static analysis for state-space reduction of polygonal hybrid systems. In: Asarin, E., Bouyer, P. (eds.) FORMATS 2006. LNCS, vol. 4202, pp. 306–321. Springer, Heidelberg (2006)

10. Pace, G., Schneider, G.: Model checking polygonal differential inclusions using invariance kernels. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 110–121. Springer, Heidelberg (2004)
11. Asarin, E., Pace, G., Schneider, G., Yovine, S.: SPeeDI: a verification tool for polygonal hybrid systems. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 354–358. Springer, Heidelberg (2002)
12. Pace, G., Schneider, G.: A compositional algorithm for parallel model checking of polygonal hybrid systems. In: Barkaoui, K., Cavalcanti, A., Cerone, A. (eds.) ICTAC 2006. LNCS, vol. 4281, pp. 168–182. Springer, Heidelberg (2006)