HyPro: A C++ Library of State Set Representations for Hybrid Systems Reachability Analysis*

S. Schupp[1], E. Ábrahám[1], I. B. Makhlouf[1], and S. Kowalewski[1]

RWTH Aachen University, Germany

Abstract. In this tool paper we introduce HyPro, our free and open-source C++ programming library, which offers implementations for the most prominent state set representations used by flowpipe-construction-based reachability analysis techniques for hybrid systems.

1 Introduction

As *hybrid systems* with mixed discrete-continuous behaviour are often safety-critical applications, a rising interest in their safety verification resulted in the development of powerful tools implementing different approaches to determine the set of system states that are reachable from a given set of initial states. Besides approaches based on, e.g., theorem proving or SMT solving, *flowpipe-construction-based reachability analysis* is a well established method, which over-approximates the set of reachable states of a hybrid system by a union of state sets, each of them being *represented* by a geometric object of a certain shape (like boxes, polytopes, or zonotopes) or symbolically (like support functions or Taylor models). Hybrid systems reachability analysis tools like, e.g., Cora [1], Flow* [2], HyCreate [7], HyReach [8], SoapBox [5], and SpaceEx [3] implement different techniques using different geometric or symbolic state set representations, each of them having individual strengths and weaknesses.

The implementation of novel reachability analysis algorithms that use some geometric or symbolic state set representations is still effortful, as datatypes for the underlying state set representations need to be implemented first. In this paper we report on the first release of our free and open-source C++ library HyPro, providing implementations for the most prominent state set representations. Our aim is to offer assistance for the rapid implementation of new algorithms by encapsulating all representation-related issues and allowing the developers to focus on higher-level algorithmic aspects.

The HyPro library specifies a unified interface for different representations, which supports all operations required in reachability analysis as well as conversion methods between the different representations. Besides own implementations for state set representations, the library also offers approaches towards wrapping other existing libraries implementing a certain state set representation.

* This work was supported by the German Research Council (DFG) in the context of the HyPro project. The original publication is available at http://www.springerlink.com.

After some preliminaries in Sec. 2, we describe in Sec. 3 the structure and usage of our library and provide some experimental evaluation in Sec. 4.

2 Hybrid Systems Reachability Analysis

Reachability analysis aims at the computation of the set of states that are reachable in some system from a given set of initial states. Reachability analysis is often used for safety verification by showing that the set of reachable states does not intersect with a pre-defined set of unsafe states.

We are interested in reachability analysis for *hybrid automata* [6], a popular modelling formalism for hybrid systems. Intuitively, they extend discrete automata models, whose nodes resp. transitions model the states (*control modi*) resp. state changes (*jumps*) of the discrete part of the system, by additionally modelling the evolution of continuous quantities (*flowpipe*) between discrete state changes through ordinary differential equation (ODE) systems.

As the reachability problem for hybrid automata is in general undecidable, *over-approximative bounded reachability analysis* can be used to over-approximate reachability along such paths that satisfy some upper bounds on the time elapse between two consecutive jumps (*time horizon*) and on the number of jumps (*jump depth*). Due to the over-approximation, we can prove bounded safety in case of an empty intersection of the reachable state set with the unsafe state set, but no conclusive answer can be given if this intersection is not empty.

Flowpipe-construction-based reachability analysis approaches iteratively compute successors of a given initial state set. To over-approximate flowpipes, they divide a given time horizon into time segments and over-approximate the states reachable within each time segment by a state set, thus "paving" the flowpipe with state sets. For computing jump successors, they determine the intersections of those "paving" state sets with the guards of jumps that exit the current control modus, and apply the jumps' reset transformations to those intersections (see Fig. 1).

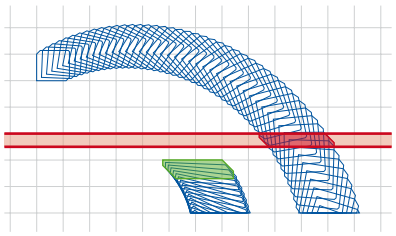

Fig. 1. Flowpipe-construction-based reachability analysis (guard satisfying sets in red, jump successor in green).

3 The HyPro Library

The library is published at https://github.com/hypro/hypro. In the following we describe its components (see Fig. 2) and its usage. For more details we refer to the online documentation and the user's guide accessible on the above page.

Arithmetic computations HyPro is templated in the number type and makes use of `boost` and the following external libraries:

- `cln`, GMP (optional): exact number types `cl_RA` and `mpq_class`;

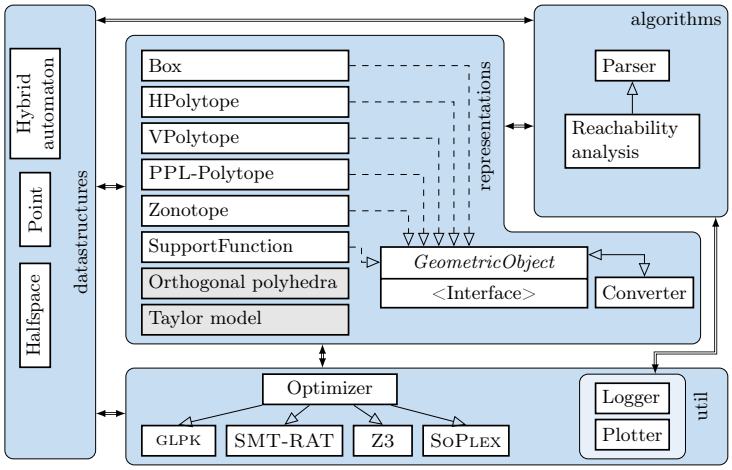

Fig. 2. HyPro class structure.

- CArL: number-type-templated (`cl_RA` or `mpq_class`) exact arithmetic computations, number type conversion;
- Eigen3: number-type-templated matrix computations; when instantiated with `double`, conservativeness is not assured;
- PPL (optional): efficient but inexact computations with polytopes;
- glpk: linear optimiser using either floating-point or exact arithmetic, however, its interface does not support the exchange of exact numbers, thus the results are not provably correct;
- SMT-RAT, SoPlex and Z3 (optional): exact linear optimisers; SMT-RAT and SoPlex support `mpq_class` in their interfaces, but not Z3, therefore we need to convert `mpq_class`-numbers to strings when calling Z3;
- `log4cplus` (optional): logger functionalities.

Currently, HyPro can be instantiated with inexact (`double`) or exact (`cl_RA`, `mpq_class`) number types; Eigen3 will be instantiated the same way. When *inexact*, all representations as well as Eigen3 use the `double` number type, thus we cannot guarantee over-approximative results; however, as exact optimisation is extremely important for meaningful results for most representations, we still guarantee exact optimisation through a combination of inexact glpk with an exact optimiser if available (see Fig. 3). When using an *exact* number type, HyPro assures conservative results if one of the modules SMT-RAT, SoPlex or Z3 are available and if PPL is not used; as glpk is faster than the other optimisers but its interface is inexact, we use the same approach as for the `double` representation shown in Fig. 3, but run glpk in exact modus.

State set representations To implement the computations described in the previous section, we need a suitable data type (*representation*) that supports the *storage* of state sets (subsets of $\mathbb{R}^n$) and certain *operations* on them. The choice of the state set representation is highly relevant, as it strongly influences both

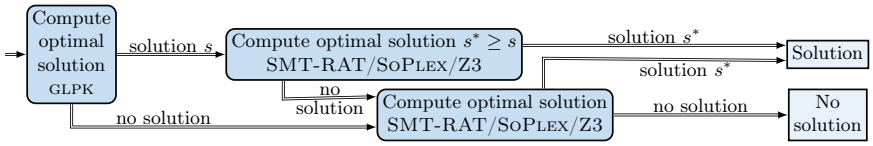

Fig. 3. Increased efficiency by combining inexact and exact computations.

computational effort and precision. Our library offers state-set representation by *boxes*, *(convex) polytopes* [10] in vertex ($\mathcal{V}$) as well as in halfspace ($\mathcal{H}$) representation, *support functions* [9] and *zonotopes* [4]. For these representations, we provide all operations needed for the reachability analysis of *linear* hybrid automata (hybrid automata specified using linear conditions and resets, and linear ODEs): linear transformation, Minkowski sum, intersection, union, and test for emptiness. All the above representations implement a common interface specifying these operations, extended with some additional convenience functions (e.g., functions for determining the dimension of a set or functionalities for output). Some representations also extend this interface with individual functions, only relevant for that representation (e.g., order reduction functions for zonotopes).

We additionally provide a module for *orthogonal polyhedra*, but it is partial as we found no proper way to compute the Minkowski sum and linear transformation. We thank Xin Chen who contributed with a further module for *Taylor models*; however, as Taylor-model-based reachability analysis requires different operations, this module does not implement the global HyPro interface.

Conversion None of the state set representations is generally optimal in terms of both computational effort and precision in reachability analysis. Switching between representations, although mostly expensive, can pay off during the analysis, for instance to improve the precision of the computed state sets locally. This feature allows for the implementation of backtracking mechanisms and fast look-ahead strategies in a dynamic reachability analysis approach. HyPro implements easy-to-use (exact or over-approximating) conversion operations for all included state set representations; this converter is a template parameter and thus exchangeable by the user, if more specialised methods are desired.

Reduction techniques The size of state set representations usually strongly increases during the analysis due to more complex shapes (e.g., when computing Minkowski sum) and number representations (e.g., when computing linear transformation). For boxes and polytopes, HyPro provides efficient and conservative over-approximating number reduction techniques. For zonotopes we offer a conservative order-reduction algorithm to limit the number of generators. For support functions we reduce the operational tree of the object.

Additional datastructures and utility functions We provide a data type for hybrid automata, a parser for Flow*-like syntax, utility functions such as a plotter which creates gnuplot or TikZ output files for state set visualisation, logging mechanisms to trace executions, and an exemplary reachability analysis algorithm among various other examples showing how to use the library.

Usage We illustrate the usage of the HyPro library on some simple examples based on the `double` number type (where also Eigen3 objects are instantiated with `double`); for further details see the `examples` folder and the user's guide.

We can create a state set $\{x \in \mathbb{R}^n | Ax \leq b\}$ represented by an $\mathcal{H}$-polytope `p` by specifying an Eigen3 matrix `A`, representing the constraints (row-wise) and an Eigen3 vector `b` representing the constant parts, as follows:

```
HPolytope<double> p = HPolytope<double>(A, b);
```

The Minkowski sum `p` of two $\mathcal{H}$-polytopes `p1` and `p2` can be computed by:

```
HPolytope<double> p = p1.minkowskiSum(p2);
```

A box containing a set `V` of points of type `std::vector<Point<double>>` can be converted to a polytope in the $\mathcal{H}$-representation using the `Converter` class:

```
HPolytope<double> p = Converter::toHPolytope(Box<double>(V));
```

To plot an object (per default in the first two dimensions), we can report its vertices to the singleton class `Plotter`, and create a gnuplot file using the method `plot2d()`:

```
Plotter<double>::getInstance().addObject(p.vertices());
Plotter<double>::getInstance().plot2d();
```

Future work Currently we focus on efficiency-related improvements for the presented representations, including the better exploitation of inexact arithmetic. Long-term plans address also extensions with further representations. Regarding efficiency, naturally, we cannot compete with well-established special-purpose libraries like PPL and polymake for polytope computations. Additionally to PPL, we work on the development of further wrappers for third-party libraries. Last but not least, as representation-related parameter settings are currently global and static, we work on the support of representation- and object-specific settings.

4 Experimental Evaluation

Using our library we implemented a simple reachability analysis algorithm for linear hybrid systems, and used it to evaluate the efficiency of our library on three commonly known benchmarks: (1) the *bouncing ball* (BBall) models the bouncing of an elastic ball dropped from a predefined height (parameters: time step $\delta = 0.01$, time horizon $T = 3$); (2) the *rod reactor* (Rod) models the temperature controller of a nuclear power plant and its cooling dynamics ($\delta = 0.01$, $T = 17$); (3) the *switching 5D linear system* (5D SW) is an artificially created benchmark in 5 dimensions with planar guards ($\delta = 0.001$, $T = 0.2$).

All experiments were carried on an Intel Core i7 (4×4 GHz) CPU with 16 GB RAM. Tab. 1 shows the results when using `mpq_class` (exact) and `double` (inexact) number types, and as representations boxes (Box), $\mathcal{H}$-polytopes (HPoly), $\mathcal{V}$-polytopes which are converted to $\mathcal{H}$-polytopes for intersection computation (VPoly), polytope representation by the PPL library (PPL), support functions (SF) and zonotopes (Zono). For both `mpq_class` and `double`, we distinguish glpk only in exact resp. inexact modus, and `glpk`+SMT-RAT and `glpk`+Z3 combining glpk with an exact solver as in Fig. 3. Inexact-arithmetic results that

		mpq_class			double			SpaceEx	
		GLPK	GLPK+ SMT-RAT	GLPK+ Z3	GLPK	GLPK+ SMT-RAT	GLPK+ Z3	LGG	STC
Box	BBall	0.1	0.1	0.1	0.002	0.002	0.03	0.003	0.01
	Rod	63.8	64.8	65.1	0.01	0.06	0.02	0.02	0.2
	5D SW	0.3	0.3	0.3	0.02	0.02	0.02	0.02	0.03
HPoly	BBall	1.2	1.1	8.7	0.2	0.7	4.9	-	-
	Rod	24.3	21.3	136.5	4.8	16.1	131	-	-
	5D SW	54.8	TO	TO	4.3	TO	TO	-	-
VPoly	BBall	1.8	1.5	6.0	TO	(0.7)	(5.5)	-	-
	Rod	100.2	98.7	171.5	TO	(0.3)	(2.6)	-	-
	5D SW	TO	TO	TO	TO	TO	TO	-	-
PPL	BBall	0.07	0.07	0.08	0.05	0.06	0.06	-	-
	Rod	2.7	2.6	2.9	1.8	1.9	1.9	-	-
	5D SW	TO	TO	TO	TO	TO	TO	-	-
SF	BBall	0.6	2.0	15.6	0.02	1.1	43.8	0.2	0.03
	Rod	72.8	101.6	1125.8	0.4	54.4	609.6	1.1	0.9
	5D SW	270.6	279.8	411.1	0.04	2.6	319.3	0.8	0.2
Zono	BBall	TO	TO	TO	0.006	0.007	0.006	-	-
	Rod	4.8	4.9	4.9	0.02	0.02	0.02	-	-
	5D SW	3.8	3.9	3.9	0.004	0.004	0.004	-	-

Table 1. Benchmark results with runtimes in seconds (*TO* for $\geq$ 20 minutes). Dashes indicate that a tool does not support this kind of state set representation.

we (manually) detected to be under-approximating are put in parenthesis; this occurred for VPoly due to inexact Eigen3 computations. For comparison, we present SpaceEx results using support functions (SF) as well as SF with box templates (Box); note that SpaceEx uses `double` representation and glpk.

Due to space limitation, we discuss only some timing issues. At least on these few examples, HyPro in inexact glpk-only modus is competitive with SpaceEx. A higher computational effort can be observed for exact arithmetic, most prominently for SF, which highly relies on optimisation; the longer running times for glpk +Z3 (wrt. SMT-RAT) are due to the string-based interface communication overhead. For 5D SW, the initial set is a single point. Zonotopes, performing well on small initial sets, deliver very good results here.

References

1. Althoff, M., Dolan, J.M.: Online verification of automated road vehicles using reachability analysis. IEEE Transaction on Robotics 30(4), 903–918 (2014)
2. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: Proc. CAV'13. LNCS, vol. 8044, pp. 258–263. Springer (2013)
3. Frehse, G., Guernic, C.L., Donzé, A., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable verification of hybrid systems. In: Proc. CAV'11. LNCS, vol. 6806, pp. 379–395. Springer (2011)
4. Girard, A.: Reachability of uncertain linear systems using zonotopes. In: Proc. HSCC'05. LNCS, vol. 3414, pp. 291–305. Springer (2005)
5. Hagemann, W., Möhlmann, E., Rakow, A.: Verifying a PI controller using SoapBox and Stabhyli: Experiences on establishing properties for a steering controller. In: Proc. ARCH'14. EPiC Series in Computer Science, vol. 34. EasyChair (2014)

6. Henzinger, T.: The theory of hybrid automata. In: Proc. of LICS'96. pp. 278–292. IEEE Computer Society Press (1996)
7. HyCreate. http://stanleybak.com/projects/hycreate/hycreate.html
8. HyReach. https://embedded.rwth-aachen.de/doku.php?id=en:tools:hyreach
9. Le Guernic, C., Girard, A.: Reachability analysis of linear systems using support functions. Nonlinear Analysis: Hybrid Systems 4(2), 250–262 (2010)
10. Ziegler, G.M.: Lectures on polytopes, vol. 152. Springer Science & Business Media (1995)