Linear Hybrid System Falsification through Local Search*

Houssam Abbas and Georgios Fainekos

Arizona State University, Tempe, AZ, USA
{hyabbas,fainekos}@asu.edu

Abstract. In this paper, we address the problem of local search for the falsification of hybrid automata with affine dynamics. Namely, given a sequence of locations and a maximum simulation time, we return the trajectory that comes closest to the unsafe set. This problem is formulated as a differentiable optimization problem and solved. The purpose of developing such a local search method is to combine it with high level stochastic optimization algorithms in order to falsify hybrid systems with complex discrete dynamics and high dimensional continuous spaces. Experimental results indicate that the local search procedure improves upon the results of pure stochastic optimization algorithms.

Keywords: Model Validation and Analysis; Robustness; Simulation; Hybrid systems.

1 Introduction

Despite the recent advances in the computation of reachable sets in medium to large-sized linear systems (about 500 continuous variables) [1], the verification of hybrid systems through the computation of the reachable state space remains a challenging problem [2]. To overcome this difficult problem, many researchers have looked into testing methodologies as an alternative. Testing methodologies can be coarsely divided into two categories: robust testing (e.g. [3, 4] and systematic/randomized testing [5, 6].

Along the lines of randomized testing, we investigated the application of Monte Carlo techniques [7] to the temporal logic falsification problem of hybrid systems. In detail, utilizing the robustness of temporal logic specifications [8] as a cost function, we managed to convert a decision problem, i.e., does there exist a trajectory that falsifies the system, into an optimization problem, i.e., what is the trajectory with the minimum robustness value? The resulting optimization problem is highly nonlinear and, in general, without any obvious structure. Therefore, we treated the model of the hybrid system as a black box, and the cost function was minimized using Simulated Annealing (SA).

* This work was partially supported by a grant from the NSF Industry/University Cooperative Research Center (I/UCRC) on Embedded Systems at Arizona State University and NSF award CNS-1017074.

T. Bultan and P.-A. Hsiung (Eds.): ATVA 2011, LNCS 6996, pp. 503–510, 2011.

A stochastic optimization algorithm for the falsification problem picks a point in the set of initial conditions, simulates the system for a bounded duration, computes the distance to the unsafe set and, then, decides on the next point in the set of initial conditions to try. Our goal in this paper is to provide assistance at exactly this last step. Namely, how do we pick the next point in the set of initial conditions? Note that we are essentially looking for a descent direction for the cost function in the set of initial conditions.

Our main contribution, in this paper, is an algorithm that can propose such descent directions. Given a test trajectory $s_{x_0} : \mathbb{R}_+ \mapsto \mathbb{R}^n$ starting from a point x_0, the algorithm tries to find some vector d such that s_{x_0+d} gets closer to the unsafe set than s_{x_0}. We prove that it converges to a local minimum of the robustness function in the set of initial conditions, and demonstrate its advantages within a stochastic falsification algorithm. These results will enable local descent search for the satisfaction of arbitrary linear temporal logic specifications, not only safety specifications. The extended version of the paper appears in [9].

2 Problem Formulation

The results in this paper will focus on the model of hybrid automata with affine dynamics. A hybrid automaton is a mathematical model that captures systems that exhibit both discrete and continuous dynamics. In brief, a *hybrid automaton* is a tuple $\mathcal{H} = (X, L, E, Inv, Flow, Guard, Re)$ where $X \subseteq \mathbb{R}^n$ is the state space of the system, L is the set of control locations, $E \subseteq L \times L$ is the set of control switches, $Inv : L \to 2^X$ assigns an invariant set to each location, $Flow : L \times X \to \mathbb{R}^n$ defines the time derivative of the continuous part of the state, $Guard : E \to 2^X$ is the guard condition that enables a control switch e and, $Re : X \times E \to X \times L$ is a reset map. Finally, we let $H = L \times X$ to denote the state space of the hybrid automaton $\mathcal{H}$. For the purposes of this paper, we define a *trajectory* η_{h_0} starting from a point $h_0 \in H$ to be a function $\eta_{h_0} : \mathbb{R}_+ \to H$ defined by: $\eta_{h_0}(t) = (l(t), s_{x_0}(t))$, where $l(t)$ is the location at time t, and $s_{x_0}(t)$ is the continuous state at time t. We will denote by $\mathrm{loc}(\eta_{h_0})$ the sequence of control locations that the trajectory η_{h_0} visits (no repetitions).

The hybrid systems dealt with in this paper are deterministic and non-Zeno. In each location, the dynamics are affine, the guards are non-overlapping and the transitions are taken as soon as possible. This will permit us to use directly results from [4]. To avoid a digression into unnecessary technicalities, we will assume that the set of initial conditions $X_0 \subset \mathbb{R}^n$ and the unsafe set $\mathcal{U} \subset H$ are included in single control locations, l_0 and $l_{\mathcal{U}}$, respectively.

Let $D_{\mathcal{U}} : H \mapsto \mathbb{R}_+$ be the distance function to $\mathcal{U}$, defined by $D_{\mathcal{U}}(v, x) = \inf_{u \in \mathcal{U}} ||x - u||$ if $v = l_{\mathcal{U}}$, and $D_{\mathcal{U}}(v, x) = +\infty$ otherwise. Given a compact time interval $[0, T]$, we define the *robustness* of a system trajectory η_h to be $f(h) \triangleq \min_{0 \le t \le T} D_{\mathcal{U}}(\eta_h(t))$. When l is clear from the context, we'll write $f(x)$. Trajectories of minimal robustness indicate potentially unsafe operation of the system. Finding such a trajectory can be seen as a 2-stage problem: first, decide on a sequence of locations to be followed by the trajectory. Second, out of

all trajectories following this sequence of locations, find the trajectory of minimal robustness. This paper addresses the second stage. The central step is the solution the following problem:

Problem 1 *Given a hybrid automaton $\mathcal{H}$, a compact time interval $[0, T]$, a set of initial conditions $H_0 \subseteq H$ and a point $h_0 = (l_0, x_0) \in H_0$ such that $0 < f(h_0) < +\infty$, find a vector dx such that $h'_0 = (l_0, x_0 + dx)$, $loc(\eta_{h_0}) = loc(\eta_{h'_0})$ and $f(h'_0) \leq f(h_0)$.*

An efficient solution to Problem 1 may substantially increase the performance of the stochastic falsification algorithms by proposing search directions in which the robustness decreases. In summary, our contributions are: a) We formulate Problem 1 as a nonlinear optimization problem, which we prove to be differentiable w.r.t. the initial conditions. Thus it is solvable with standard optimizers. b) We developed an algorithm, Algorithm 1, to find local minima of the robustness function. c) We demonstrate the use of Algorithm 1 in a higher-level stochastic falsification algorithm, and present experimental results to analyze its competitiveness against existing methods. We now make some **assumptions**:

a. The continuous dynamics in each location are stable.[1]
b. The resets $Re(\cdot, e)$ are differentiable in their first argument.
c. Conditions 4 and 5 of Theorem III.2 in [12] are satisfied, namely: for all i, there exists a differentiable function $\sigma_i : \mathbb{R}^n \mapsto \mathbb{R}$ such that $Inv(l_i) = \{x \in \mathbb{R}^n | \sigma_i(x) \geq 0\}$; and, for all i, x such that $\sigma_i(x) = 0$, the Lie derivative $L_F \sigma_i(x) \neq 0$. This allows us to have *differentiable* transition times t_x of the trajectory starting at the initial point $x \in X_0$.
d. $l_{\mathcal{U}} \in loc(\eta_{h_0})$. This is required for our problem to be well-defined (specifically, for the objective function to have finite values). The task of finding such an h_0 is delegated to the higher-level stochastic search algorithm, within which our method is integrated. Due to space restrictions, all proofs are relegated to the technical report [9].

3 Descent in the Robustness Ellipsoid

Consider a trajectory η_{h_0} with positive robustness, with $\text{loc}(\eta_{h_0}) = l_0 l_1 \dots l_N$. This is provided by the simulation. We search for an initial point $h'_0 \in H_0$ (actually $x'_0 \in X_0$), whose trajectory gets closer to the unsafe set than the current trajectory η_{h_0}. In order to satisfy the constraints of Problem 1, we need to make sure that the new point h'_0 generates a trajectory that follows the same sequence of locations as η_{h_0}. This constraint can be satisfied using the notion of robust neighborhoods introduced in [4]. In [4], it is shown that for stable systems and for a given safe initial point $h_0 = (l_0, x_0)$, there exists an 'ellipsoid of robustness' $E(x_0)$ centered on x_0, such that any trajectory starting in the ellipsoid follows the same sequence of locations as η_{h_0}. Therefore, we restrict the

[1] This is not a restrictive assumption since we can also consider incrementally stable systems [10], and even unstable linear systems [11].

choice of initial point to $X_0 \bigcap E(x_0)$, where $E(y) = \{x|(x-y)^T P_y^{-1}(x-y) \leq 1\}$ is the ellipsoid of robustness centered on y, with shape matrix P_y.

We now proceed to pose our search problem as a feasibility problem. Let t_0 be the time at which s_{x_0} is closest to $\mathcal{U}$, and $\mathcal{W}$ be the set of all points which are closer to $\mathcal{U}$ than $s_{x_0}(t_0)$. $\mathcal{W}$ is represented as $\mathcal{W} = \{x \in \mathbb{R}^n : p_i(x) \leq 0, i = 1 \dots k\}$, where the p_i are suitably defined predicates, and $X_0 = \{x|C_0 x - g_0 \leq 0\}$. If there exists $x^* \in X_0 \bigcap E(x_0)$ and $t^* \geq 0$ such that $s_{x^*}(t^*) \in \mathcal{W}$, it follows that $f(x^*) \leq f(x_0)$. Our search problem then consists in finding such x^* and t^*. Therefore define the decision variable $z = (x, t, \nu) \in \mathbb{R}^n \times \mathbb{R}_+ \times \mathbb{R}$, the objective function $F(z) = \nu$, and the constraint functions: $G_0(z) = C_0 x - g_0$, $G_E(z) = (x - x_0)^T P_{x_0}^{-1}(x - x_0) - 1$, and $G_{\mathcal{W}}(z) = (p_1(s_x(t)), \dots, p_k(s_x(t)))^T$. The search problem can now be cast as a feasibility problem over z:

$$\min_{z=(x,t,\nu)} F(z) \text{ s.t. } G_0(z) \leq 0, G_E(z) \leq \nu, G_{\mathcal{W}}(z) \leq \nu \tag{1}$$

(In our implementation of Problem (1), the first constraint is specified as bounds to the optimization and so is always satisfied).

The objective function $F(z)$ measures the *slack* in satisfying the constraints: a negative ν means all constraints are satisfied, and in particular, $G_{\mathcal{W}}$. Thus, we have a trajectory that enters $\mathcal{W}$ and, hence, gets closer to $\mathcal{U}$. Formally:

Proposition 1. *Let $z^* = (x^*, t^*, \nu^*)$ be a minimum of $F(z)$ in program (1). Then $f(l_0, x^*) \leq f(l_0, x_0)$.*

Functions F, G_0 and G_E are differentiable in z. The next proposition asserts differentiability of $G_{\mathcal{W}}$. Thus, standard gradient-based optimizers can be used to solve Problem (1). Let $E_0 \triangleq \text{int}(E(X_0) \bigcap X_0)$.

Proposition 2. *Fix $t \in (t_{N-1}, T]$, and consider the hybrid trajectory over $N \geq 1$ locations. Then $s_x(t)$ is differentiable at x_0 for all $x_0 \in E_0$. Moreover, for a fixed $x \in E_0$, $s_x(t)$ is differentiable in t over (t_{N-1}, T). If p_i is differentiable for all $i = 1, \dots, k$, then $G_{\mathcal{W}}$ is differentiable in z.*

We choose Sequential Quadratic Programming (SQP), as a good general-purpose optimizer to solve Problem 1. SQP is a Q-quadratically convergent iterative algorithm. At each iterate, $G_{\mathcal{W}}(x_i, t_i, \nu_i)$ is computed by simulating the system at x_i. This is the main computational bottleneck of this method, and will be discussed in more detail in the Experiments section.

Solving Problem (1), for a given $\mathcal{W}$, produces a descent direction for the robustness function, but not necessarily a minimum. Algorithm 1 (RED) describes how to setup a *sequence* of optimization problems that leads to a local minimum of f (see [9] for proof): for $i = 0, 1, 2, \dots$, let $x_i \in X_0 \bigcap E(x_{i-1})$, and let t_i be the time when s_{x_i} is closest to $\mathcal{U}$. Let $\mathcal{W}_i$ be the set of points closer to $\mathcal{U}$ than $s_{x_i}(t_i)$. For each $\mathcal{W}_i$, one can setup the optimization Problem (1) with $\mathcal{W} = \mathcal{W}_i$, and initial point $(x_i, t_i, 0)$; this problem is denoted by Prob1[$\mathcal{W}_i$].

Ellipsoid Descent with Stochastic Falsification: As outlined in the introduction, RED can be used as a sub-routine in a higher-level stochastic search

Algorithm 1. Robustness Ellipsoid Descent (RED)

Input: An initial point $x_0 \in X_0$, and corresponding t_0.
Output: z_Q.

1: Initialization: $i = 0$
2: Compute $z_i^* = (x_i^*, t_i^*, \nu_i^*) =$ minimum of Prob1$[\mathcal{W}_i]$.
3: **while** $\nu_i^* < 0$ **do**
4: $\quad x_{i+1} \leftarrow x_i^*$
5: $\quad t_{i+1} = \arg\min_t d_{\mathcal{U}}(s_{x_{i+1}}(t))$
6: $\quad \mathcal{W}_{i+1} = P(x_{i+1})$
7: $\quad$ Compute $z_i^* = (x_i^*, t_i^*, \nu_i^*) =$ min of Prob1$[\mathcal{W}_{i+1}]$.
8: $\quad i = i + 1$
9: **end while**
10:
11: Return $z_Q \triangleq z_i^*$

Algorithm 2. Simulated Annealing with RED (SA+RED)

Input: An initial point $x \in X_0$.
Output: Samples $\Theta \subset X_0$.
Initialization: BestSoFar $= x$, $f_b = f$(BestSoFar)

1: **while** $f(x) > 0$ **do**
2: $\quad x' =$ ProposalScheme(x)
3: $\quad \alpha = \exp(-\beta(f(x') - f_b))$
4: $\quad$ **if** $UniformRandom(0,1) \leq \alpha$ **then**
5: $\quad\quad x =$ RED(x')
6: $\quad$ **else**// Use the usual acceptance criterion
7: $\quad\quad \alpha = \exp(-\beta(f(x') - f(x)))$
8: $\quad\quad$ **if** $UniformRandom(0,1) \leq \alpha$ **then** $x = x'$
9: $\quad\quad$ **end if**
10: $\quad$ **end if**
11: $\quad$ (BestSoFar,f_b) = BetterOf(x, BestSoFar)
12: **end while**

falsification algorithm. A stochastic search will have a ProposalScheme routine which, given a point x in the search space, will propose a new point x' as a falsification candidate. RED may then be used to further descend from some judiciously chosen proposals. Algorithm 2 illustrates the use of RED within the Simulated Annealing (SA) stochastic falsification algorithm of [7]. Given two samples x and y, BetterOf(x, y) returns the sample with smaller robustness, and its robustness.

For each proposed sample x', it is *attempted* with certainty if its robustness is less than the smallest robustness f_b found so far. Else, it is attempted with probability $e^{-\beta(f(x')-f_b)}$ (lines 3-4). If x' is attempted, RED is run with x' as starting point, and the found local minimum is used as final accepted sample (line 5). If the proposed sample is not attempted, then the usual acceptance-rejection criterion is used: accept x' with probability $\min\{1, e^{-\beta(f(x')-f(x))}\}$. As in the original SA method, ProposalScheme is implemented as a Hit-and-Run

sampler (other choices can be made). The next section presents experimental results on three benchmarks.

3.1 Experiments

This section describes the experiments used to test the proposed algorithm SA+RED. The technical report [9] contains details of the benchmarks, methods, experiments and more results. We chose 3 navigation benchmarks from the literature: Nav0 (4-dimensional with 16 locations, unknown whether it is falsifiable or not), and Nav1 and Nav2 (4-dimensional with 3 locations, both falsifiable); and a filtered osciallator Fosc (32-dimensional with 4 locations). The methods compared are: SA+RED, pure Simulated Annealing (SA) [7], and the reachability analysis tool SpaceEx [13]. In a falsification framework, SpaceEx is used as follows: for a given bound j on the number of discrete jumps, SpaceEx computes an *over*-approximation $\overline{R(j)}$ of the set $R(j)$ reachable in j jumps: $R(j) \subset \overline{R(j)}$. If $\overline{R(j)} \cap \mathcal{U}$ is empty, then *a fortiori* $R(j) \cap \mathcal{U}$ is empty, and the system is safe if trajectories are restricted to j jumps. When, however, $\overline{R(j)} \cap \mathcal{U} \neq \emptyset$, no conclusion can be drawn. Because SA and SA+RED are stochastic methods, their behavior will be studied by analyzing a number of runs. A *regression* will mean a set of 20 runs, all executed with the same set of parameters, on the same benchmark. SpaceEx was run in deterministic mode on Nav0.

Parameter setting: We set the test duration $T = 12$sec for all benchmarks. For SA+RED, we chose to generate 10 samples ($|\Theta| = 10$). Even this small number is enough for the algorithm to be competitive. The SpaceEx parameters were varied in such a way that the approximation $\overline{R}$ of the reachable set R became increasingly precise. See [9].

The performance and cost metrics: Each run produces a minimum robustness. For a given regression, we measure: the smallest, the average, and the largest minimum robustness found by the regression (min, avg, max in Table 1). The standard deviation of minimum robustness is also reported (σ_f). For SpaceEx, we had to simply assess whether $\overline{R(j)}$ intersected $\mathcal{U}$ or not. SA and SA+RED each simulates trajectories of a fixed length T in the course of its operation, so their costs are compared by looking at the average Number of Trajectories ($\overline{NT}$) each simulates. The operations that SpaceEx does are radically different from those of the other methods compared here. The only way to compare performance is through the runtime.

Experiments: We impose an upper limit NT_{MAX} on NT: SA+RED is aborted when its NT reaches this maximum, and SA is made to generate NT_{MAX} samples. (Of course, SA+RED might converge before simulating all NT_{MAX} trajectories). 3 values were chosen for NT_{MAX}: 1000, 3000 and 5000. For each value, a regression is run and the results reported.

Table 1 compares SA+RED to SA: we start by noting that SA+RED falsified Nav2, whereas SA failed to so. On most regressions, SA+RED achieves better performance metrics than SA, for the same (or lower) computational cost. This is

Table 1. Comparison of SA and SA+RED. Robustness values are reported to the first differing decimal at least. σ_f is standard deviation of robustness for SA+RED.

System	NT_{MAX}	$\overline{NT}$ (σ_{NT})	σ_f	SA+RED Rob. min, avg, max	SA Rob. min, avg, max
Nav0	1000	1004 (1.4)	0.022	0.2852, 0.30,0.35	0.2853,0.33,0.33
	3000	2716 (651)	0.019	0.2852,0.29,0.32	0.2858,0.31,0.36
	5000	4220 (802)	0.009	0.285,0.28,0.32	0.286,0.32,0.35
Nav1	1000	662 (399)	0.21	0,0.43,0.65	0,0.96,1.88
	3000	1129 (1033)	0.23	0,0.39,0.65	0,0.99,1.80
	5000	1723 (1770)	0.23	0,0.38,0.68	0,0,0
Nav2	1000	902 (246)	0.32	0,0.54,0.78	0.3089,1.11,1.90
	3000	1720 (1032)	0.3	0,0.53,0.83	0.3305,1.29,1.95
	5000	1726 (1482)	0.27	0,0.62,0.79	0,0.002,0.01
Fosc	1000	1000 (9.3)	0.024	0.162,0.206,0.251	0.1666,0.216,0.271
	3000	3000 (8.7)	0.024	0.163,0.203,0.270	0.173,0.212,0.254
	5000	5000 (11)	0.028	0.167,0.193,0.258	0.185, 0.218, 0.245

consistent whether considering best case (min), average case (avg) or worst case (max). There are 2 exceptions: on Nav1 and Nav2, $NT_{MAX} = 5000$ produces better average and max results for SA than for SA+RED. When running realistic system models, trajectory simulation is the biggest time consumer, so effectively NT is the limiting factor. So we argue that these 2 exceptions don't invalidate the superiority of SA+RED as they occur for high values of NT that might not be practical with real-world models.

For SpaceEx running on Nav0, we observed that our initial parameter set produces an $\overline{R(j)}$ that intersects $\mathcal{U}$. Since this is inconclusive, we modified the parameters to get a better approximation, but SpaceEx runtimes far exceeded those of SA+RED (more than 1.5 hours). Moreover, SpaceEx did not reach a fixed point of its iterations. Thus, we can not be sure that all of the reachable space was covered. While this may be seen as an analogous problem to the choice of T in SA+RED, the computational cost of increasing j is much more prohibitive than that of increasing T. Thus we may conclude that stochastic falsification and reachability analysis can play complementary roles in good design practice: first, stochastic falsification computes the robustness of the system with respect to some unsafe set. Guided by this, the designer may make the system more robust, which effectively increases the distance between the (unknown) reachable set and the unsafe set. Then the designer can run a reachability analysis algorithm where coarse over-approximations can yield conclusive results.

4 Conclusions

The minimum robustness of a hybrid system is an important indicator of how safe it is. In this paper, we presented an algorithm for computing a local minimum of the robustness for a certain class of linear hybrid systems. When integrated with a higher-level stochastic search algorithm, the proposed algorithm has been shown to perform better than existing methods on literature benchmarks, and

to complement reachability analysis in falsification. We will next deploy this capability to perform local descent search for the satisfaction of arbitrary linear temporal logic specifications, not only safety specifications. It will be important to reduce the required number of tests NT, and to determine an appropriate test duration T, rather than a fixed arbitrary value. Finally, it is important to get a theoretical understanding of the behavior of the two Markov chains iterated by SA+RED to further improve it.

References

1. Girard, A., LeGuernic, C.: Efficient reachability analysis for linear systems using support functions. In: IFAC World Congress, pp. 22–35 (2008)
2. Althoff, M., Stursberg, O., Buss, M.: Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. Nonlinear Analysis: Hybrid Systems 4(2), 233–249 (2010)
3. Girard, A., Pappas, G.J.: Verification using simulation. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 272–286. Springer, Heidelberg (2006)
4. Julius, A.A., Fainekos, G., Anand, M., Lee, I., Pappas, G.: Robust test generation and coverage for hybrid systems. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 329–342. Springer, Heidelberg (2007)
5. Branicky, M., Curtiss, M., Levine, J., Morgan, S.: Sampling-based planning, control and verification of hybrid systems. IEE Proc.-Control Theory Appl. 153(5), 575–590 (2006)
6. Zuliani, P., Platzer, A., Clarke, E.M.: Bayesian statistical model checking with application to simulink/stateflow verification. In: Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, pp. 243–252 (2010)
7. Nghiem, T., Sankaranarayanan, S., Fainekos, G., Ivancic, F., Gupta, A., Pappas, G.: Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In: Hybrid Systems: Computation and Control (2010)
8. Fainekos, G., Pappas, G.: Robustness of temporal logic specifications for continuous-time signals. Theoretical Computer Science 410(42), 4262–4291 (2009)
9. Abbas, H., Fainekos, G.: Linear hybrid system falsification through descent, technical Report arXiv:1105.1733 (2011)
10. Tabuada, P.: Verification and Control of Hybrid Systems: A Symbolic Approach. Springer, Heidelberg (2009)
11. Julius, A.A., Pappas, G.J.: Trajectory based verification using local finite-time invariance. In: Majumdar, R., Tabuada, P. (eds.) HSCC 2009. LNCS, vol. 5469, pp. 223–236. Springer, Heidelberg (2009)
12. Lygeros, J., Johansson, K.H., Simic, S.N., Zhang, J., Sastry, S.: Dynamical properties of hybrid automata. IEEE Transactions on Automatic Control 48, 2–17 (2003)
13. Frehse, G., Guernic, C.L., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: Spaceex: Scalable verification of hybrid systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 379–395. Springer, Heidelberg (2011)