2015 19th International Conference on System Theory, Control and Computing (ICSTCC), October 14-16, Cheile Gradistei, Romania

On Probabilistic State Space Abstraction of Deterministic Switched Systems

Silvia Magdici
Department of Computer Science
Technische Universität München, Germany
E-mail: silvia.magdici@tum.de

Mircea Lazar
Department of Electrical Engineering
Eindhoven University of Technology, The Netherlands
E-mail: m.lazar@tue.nl

***Abstract*—This paper considers the problem of controlling the complexity of the state space abstraction of a deterministic switched affine system, which must satisfy a rich specification, expressed as an Linear Temporal Logic (LTL) formula. We propose a probabilistic approach to the state space abstraction problem that enables a trade-off between complexity and accuracy of the abstraction. Instead of a deterministic finite transition system ($DFTS$), the state space is abstracted to a Discrete Time Markov Chain ($DTMC$) using a regular state space partition. The transition relations between the discrete states and the corresponding probabilities are computed based on the Chebyshev radius of the intersection between one-step reachable sets and discrete states. The resulting abstraction is complete, but not minimal, i.e., it introduces some false transitions. In order to refine the abstraction, Monte Carlo Simulation is used, which yields a confidence measure for every transition, besides the assigned probability. Given the product automaton (PA) between the $DTMC$ and Büchi Automaton (BA) associated with the LTL formula, a (optimal) path generation algorithm and a controller synthesis algorithm complete the proposed solution. The application of the developed methodology to a benchmark case study from the literature, i.e., airplane fuel balancing, demonstrates the effectiveness of the approach.**

***Index Terms*—Control System Design, Hybrid Systems**

I. Introduction

Temporal Logic [1] provides a formal way to specify and verify the correctness of computer programs and digital circuits. Recently, due to its expressivity, temporal logic has been used as a specification language also in other areas than the aforementioned ones - such as robotics [2], [3], system biology [4], [5] and dynamical systems [6], [7].

In this work, we consider the following problem: *given a deterministic switched affine discrete time system and a linear temporal logic (LTL) formula ϕ over a set of linear predicates in the state space of the system, construct a state space abstraction with a predefined complexity and a control strategy such that the system trajectories satisfy the specification.* In this work, the complexity is given by the cardinality of the finite abstraction.

The classical approach to the above problem consists of two main steps. Firstly, the dynamics of the system is abstracted to a *finite transition system (FTS)* and the LTL formula is converted to a $B\ddot{u}chi$ *automaton*, e.g., using tools from [8]. Then, using e.g., the tool from [6], the satisfying paths are prescribed by the *Product Automaton* between the *finite transition system* and the $B\ddot{u}chi$ $automaton$. Observe that the tool from [6] is not dealing with probabilistic systems, but with deterministic ones. Secondly, a controller that enables a (optimal) satisfying path is designed. In what follows we recall the solutions from the literature to the state space abstraction problem, to justify our approach.

A state space abstraction method was proposed in [9] based on the following steps. Firstly, a proposition preserving partition is devised such that for each discrete state, only one atomic proposition is true within that state. Secondly, for each dynamical mode within the switched system, an additional abstraction is performed, based on backward reachability, in order to obtain a finer partition. After each dynamical mode has one list with possible transitions between the discrete states, all partitions are merged and reachability analysis is performed one more time for the merged partition. A minimal cell volume is used as a termination criterion for the refinement algorithm. Finally, a discrete planner for the deterministic transition system obtained in the first stage is synthesized; then, the planner is implemented by using a continuous controller that moves the plant between discrete states using the tool from [10]. Another method was worked out in [11], where the aim was to compute a set of initial states such that all the corresponding trajectories satisfy a given LTL formula. To this end, a dual automaton is constructed by interchanging the states and the transitions of the *Büchi Automaton*. Then, the feasibility of transitions and states for the dynamical system is checked and a further partitioning is performed based on backward and forward reachability analysis. In [12], a deterministic hybrid system is approximated with a Markov Chain in order to avoid Zeno phenomena (the condition for which, in a finite time interval, the (hybrid) trajectory jumps between specific dynamic modes infinitely many times); therein, the transitions within the Markov chain are corresponding to the transitions between discrete states of the hybrid system. Other works, see [13], [14] also report algorithms for constructing a probabilistic abstraction based on Markov Chains and use reachability analysis to assign transitions.

In this paper we also use a Markov Chain to abstract a deterministic hybrid system, as proposed in [12], [13], [14],

978-1-4799-8481-7/15/$31.00 ©2015 IEEE

and we also use reachability analysis to assign transitions in the resulting automaton, as done in [9], [13], [14]. However, differently from these previous works, we focus on controlling the complexity of the resulting abstraction. In these previous approaches, although complexity was implicitly determined by the imposed termination criterion, such as minimum volume for [9], there was no possibility to quantify a priori a desired complexity for the resulting abstraction.

The main challenge considered in this paper is therefore to develop an abstraction method that can be used for LTL control and that allows to a priori specify the desired partition complexity. This feature is particularly useful for dealing with implementation constraints present in real-life applications.

The proposed approach can be summarized by the following steps. First, a single state space regular partition is done for all dynamical modes. In our implementation we used a cubic partition, but any regular partition can be used. Then, *probabilistic* transition relations between each discrete state are assigned based on the Chebyshev radius of the intersection between one-step reachable sets and the discrete states. This results in a transition system defined by a *Discrete Time Markov Chain*. Once the transition relation is computed, the resulting abstraction is verified using Monte Carlo simulation. If the error introduced by the abstraction, compared with Monte Carlo simulation, is below a given threshold, or the partition has reached the specified complexity, the algorithm will stop. In this way, the complexity of the resulting partition can be imposed a priori, at the price of a trade off between accuracy and complexity of the resulting abstraction.

After the state space discretization and the product automaton are obtained, a path which maximizes the probability of satisfying the specification is found. Once the path is found, a *polytope-to-polytope controller* inspired from [11] is synthesized; in our case, the reachability transitions between the discrete states have some probabilities assigned, so the controller synthesis problem may be infeasible. To deal with this issue, we have developed a recovery solution for the control synthesis problem. Other, more advanced (non-deterministic) methods for controller synthesis could be employed. However, in this paper we focus on reducing complexity of the abstraction and we report a controller synthesis method for illustration purposes mainly.

Remark I.1 A probabilistic state space abstraction problem was previously considered also in [15]–[17] within the setting of stochastic hybrid systems and with a different purpose. The abstraction to a discrete time Markov chain in [15]–[17] is done based on the probabilities of the stochastic hybrid system, while our approach uses the Chebyshev Radius of the intersection between one-step reachable sets and discrete states to compute the probabilities between the states within the resulting Markov chain.

The remainder of the paper is organized as follows. In Section II we introduce some preliminaries and the problem formulation. In Section III, we present the solution - the probabilistic state space abstraction algorithm; the path generation and the control synthesis algorithms are described in Section IV. Then, in Section V, an application from the literature is considered: an aircraft fuel balancing system. Section VI concludes the paper.

II. Preliminaries and problem statement

This section contains some preliminaries on temporal logic and the problem formulation.

a) Linear Temporal Logic: Linear Temporal Logic (LTL) is an extension to classical logic, by adding temporal operators. Besides the Boolean connectors like *conjunction* $\wedge$, *disjunction* $\vee$, *implication* $\rightarrow$ and *negation* $\neg$, LTL uses also the following temporal operators: *"next"* $\bigcirc$, *"until"* $\mathcal{U}$, *"always"* $\Box$ and *"eventually"* $\Diamond$.

Let π be an atomic proposition (a predicate) given as a polyhedron (i.e., polyhedral set). A polyhedron in $\mathbb{R}^n$ is an intersection of at least $n+1$ half-spaces:

$\exists k \geq n+1, a_i \in \mathbb{R}^n, b_i \in \mathbb{R}, i = 1, \ldots, k$, such that

$$\pi = \{x \in \mathbb{R}^n \,|\, a_i^T x + b_i \leq 0, \quad \forall i = 1, \ldots, k\}. \qquad (1)$$

Given a set of atomic propositions Π, an LTL formula over Π is recursively defined as follows [14]: (i) every atomic proposition $\pi \in \Pi$ is a formula and (ii) if ϕ_1 and ϕ_2 are LTL formulas, then $\neg\phi_1$, $\phi_1 \vee \phi_2$, $\phi_1 \mathcal{U} \phi_2$ are also formulas. Formula $\phi_1 \mathcal{U} \phi_2$ means that ϕ_2 will become eventually $true$ and until that ϕ_1 is $true$. $\Diamond\phi_1$ means that ϕ_1 will eventually become $true$ and $\Box\phi_2$ indicates that ϕ_2 is always $true$ over all atomic propositions the formula is defined on. Using the entire set of LTL operators makes it possible to define complex requirements on the desired behavior of the system.

b) Problem formulation: Consider a discrete-time switched affine system described by:

$$x(t+1) = A_{\sigma(t)} x(t) + B_{\sigma(t)} u(t) + f_{\sigma(t)}, \qquad (2)$$
$$\sigma(t) : \mathbb{N} \rightarrow \mathcal{D} := \{1, \ldots, d\}; d \in \mathbb{N},$$

where $A_{\sigma(t)} \in \mathbb{R}^{n \times n}$, $B_{\sigma(t)} \in \mathbb{R}^{n \times m}$ and $f_{\sigma(t)} \in \mathbb{R}^{n \times 1}$ describe the system dynamics and $x(t) \in \mathbb{X} \subset \mathbb{R}^n$ and $u(t) \in \mathbb{U} \subset \mathbb{R}^m$ are the state and applied control at time $t \in \mathbb{N}$. The natural number d represents the number of dynamical modes. When $\sigma(t) = i$ for some $i \in \mathcal{D}$ the i^{th} subsystem is governing the dynamics (2). For brevity we will use also the notation $\{x_t\}_{t \in \mathbb{N}} = x_0 x_1 \ldots$ to denote a trajectory of the dynamical system (2), i.e., $x_t = x(t)$ for all $t \in \mathbb{N}$. In this work we assume that the switching law $\sigma(t)$ is generated by partitions of the state and input spaces $\mathbb{X}$ and $\mathbb{U}$, i.e., using the state and input trajectories $x(t)$, $u(t)$, respectively. Also, external inputs that take values in a discrete finite set may used to derive the switching law $\sigma(t)$, as illustrated in Section V.

The specification for system (2) is given as an LTL formula Φ over a set of predicates Π. A trajectory of system (2) $x_0x_1\ldots$ satisfies the formula Φ if the observed atomic propositions within the trajectory satisfy the formula. So the problem considered can be formulated as follows.

Problem II.1 Given an LTL formula Φ over a set of linear predicates Π and the dynamical system defined in (2), find a state space abstraction with an *a priori specified complexity*, a path within the abstraction for any initial condition in the state space which satisfies the formula and a feedback control strategy such that the generated path is enabled.

III. State Space Abstraction Algorithm

In this paper we abstract the state space of the system dynamics into a given number of discrete states and afterwards we provide a confidence index of the computed abstraction. To this end, the state space is abstracted into a Discrete Time Markov Chain; the partitions of the state space are regular Polyhedrons and the probabilistic transition relations are computed based on the Chebyshev radius of the intersection between one-step reachable sets and discrete states. Let us define formally a Discrete Time Markov Chain next.

Definition III.1 A Discrete Time Markov Chain is a tuple $D = (S, S_0, T_{MC}, AP, L)$, where:

- S is the set of states;
- $S_0 \subset S$ is the set of initial states;
- $T_{MC} : S \times S \to [0, 1]$ is the transition probability matrix, where $\sum_{s,s'\in S} T(s, s') = 1$;
- AP is the finite set of atomic propositions;
- $L : S \to 2^{AP}$ is a function labeling states with atomic propositions (taken from the set AP).

Define next a partition of the state-space $\mathcal{P} := \{\mathcal{P}_i\}_{i\in\{1,\ldots,N\}}$, where each cell $\mathcal{P}_i$ is a polyhedron, $\cup_{i\in\{1,\ldots,N\}}\mathcal{P}_i = \mathbb{X}$ and for all $i \neq j$ it holds that $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$. For a set $\mathcal{S}$ we use $\mathcal{S}_r$ to denote the one step reachable set from $\mathcal{S}$ corresponding to dynamics (2). Define an additional region, $\mathcal{P}_{N+1} := \mathbb{X}_r \setminus \mathbb{X}$ to account for the trajectories that start in $\mathbb{X}$ and leave it in one discrete time step. $\mathcal{P}_{N+1}$ may be different than the empty set, as we do not assume that $\mathbb{X}$ is invariant for the dynamics (2).

In this work we will use a *cubic* partitioning of the state space, for an easier point location, but any regular polytopic decomposition can be used. For each cell $\mathcal{P}_i$ with $i \neq N + 1$ the reachable set $\mathcal{P}_{r_i}$ is computed. Then, define the Chebysev radius $r_{ij} := ChebyshevRadius(\mathcal{P}_{r_i} \cap \mathcal{P}_j)$ for all $j \in \{1, \ldots, N + 1\}$. Note that the computation of the Chebysev radius can be performed automatically for any polytope (e.g., using the MPT Toolbox) and it is computationally efficient. For any cell i with $i \neq N + 1$ define the index set $\mathcal{J}_i := \{j \in \{1, \ldots, N + 1\} \,|\, \mathcal{P}_{r_i} \cap \mathcal{P}_j \neq \emptyset\}$. Then we can define the transition matrix T_{MC} for all pairs $(i, j) \in \{1, \ldots, N\} \times \mathcal{J}_i$ as follows (note that self transitions are included):

$$T_{MC}(i, j) := \frac{r_{ij}}{\sum_{l\in\mathcal{J}_i} r_{il}}.$$

Observe that $\sum_{j\in\mathcal{J}_i} T_{MC}(i, j) = 1$ for all $i \in \{1, \ldots, N\}$. For all other possible pairs (i, j) the corresponding matrix element is taken equal to zero, i.e., there is no transition in the Markov Chain from node i to node j. The abstraction of the system dynamics will not be a deterministic transition system, but a nondeterministic transition system with probabilities on transitions. The discretization algorithm using the Chebyshev radius of reachable sets is presented below in Algorithm 1.

In what follows, before stating the actual algorithm we specify the required Input data and the produced Output.

Input :

- system dynamics (2);
- input state space $\mathbb{U}$;
- $\mathbb{X}$ - state space polyhedron;
- $\mathcal{P}$ - partition of $\mathbb{X}$ and $\mathcal{P}_{N+1}$;
- $GridStep \in \mathbb{N}$ - level for state space partition.

The GridStep number is used to split each edge of the polytope $\mathbb{X}$ into equal intervals as follows: if $GridStep = l$, then each edge is partitioned into 2^l intervals. Hence, the total number of regions for $GridStep = l$ is equal to $2^l \times \ldots \times 2^l$, where the length of the product is equal to the state space $\mathbb{R}^n$ dimension. For example, in 2D for $GridStep = 1, 2, 3, 4$ we will have $4, 16, 64, 256$ cells respectively, using a uniform partitioning.

Output:

- T_{MC} - Markov chain transition matrix.

Algorithm 1 Probabilistic State Space Discretization

1: $RegularDecomposition(\mathbb{X}, GridStep)$*;
2: **for** all $i = 1, \ldots, N$ **do**
3: find the dynamics (A_l, B_l, f_l) that corresponds to $\mathcal{P}_i$;
4: $\mathcal{P}_{r_i} \leftarrow ReachableSet(\mathcal{P}_i, l, \mathbb{U}, \text{forward})$;**
5: Compute index set $\mathcal{J}_i$;
6: **for** all $j \in \mathcal{J}_i$ **do**
7: $T_{MC}(i, j) \leftarrow \frac{CR(\mathcal{P}_{r_i}\cap\mathcal{P}_j)}{\sum_{l\in\mathcal{J}_i} CR(\mathcal{P}_{r_i}\cap\mathcal{P}_l)}$;***
8: **end for**
9: **end for**
10: **return** T_{MC}

In Algorithm 1 we have used the following functions:

- $*RegularDecomposition(\mathbb{X}, GridStep)$ - recursively partitions in $GridStep$ steps the polyhedral set $\mathbb{X}$;
- $**RechableSet(\mathcal{P}_i, l, \mathbb{U}, dir)$ - computes the one-step reachable set for the polyhedral set $\mathcal{P}_i$ using affine dynamics (A_l, B_l, f_l); dir is the direction used for computing the reachable set (backward or forward); and $\mathbb{U}$ is the input state space polyhedron;

- ***$CR(\mathcal{P}_i)$ - computes the radius of the Chebysev ball of the polyhedral set $\mathcal{P}_i$.

In what follows, for brevity we will assign to each cell $\mathcal{P}_i$ a corresponding discrete state q_i. I.e., whenever the (continuous) state $x_t = x(t)$ of the dynamical system (2) lies in cell $\mathcal{P}_i$, we consider that the abstraction of system (2) has q_i as its (discrete) state. Then (q_i, q_j) is a transition in the $DTMC$ and its weight (probability) is given by $T_{MC}(i,j)$.

Next, to every sequence of continuous states $\{x_t\}_{t\in\mathbb{N}}$ that corresponds to a trajectory of the dynamical system (2), we associate a corresponding sequence of discrete states $\{q_{j_t}\}_{t\in\mathbb{N}}$ where $j_t \in \{1,\dots,N+1\}$ for all $t \in \mathbb{N}$ and which is such that $x_t \in \mathcal{P}_i$ implies $j_t = i$.

Definition III.2 The state space abstraction, given by the $DTMC$, is *complete* if all the sequences of discrete states $\{q_{j_t}\}_{t\in\mathbb{N}}$ that can be generated by trajectories $\{x_t\}_{t\in\mathbb{N}}$ of system (2) are a subset of the sequences of discrete states that can be generated by the transitions of the $DTMC$. I.e., for all $t \in \mathbb{N}$ and for any pair $(x_t, x_{t+1}) \in \mathcal{P}_i \times \mathcal{P}_j$ that satisfies (2) for some $u(t) \in \mathbb{U}$, it holds that $T_{MC}(i,j) \neq 0$.

Theorem III.3 *The finite state abstraction model ($DTMC$) constructed by Algorithm 1 is complete.*

Proof: The theorem is proven by contradiction. Let $x_0 \in \mathcal{P}_{j_0}$ for some $j_0 \in \{1,\dots,N+1\}$ be the initial state and let $x_0x_1x_2\dots$ be the corresponding trajectory generated by the dynamical system (2). By construction of the $DTMC$, we have that $x_1 \in ReachableSet(\mathcal{P}_{j_0}) \subseteq \mathcal{P}_{j_1}$ for some $j_1 \in \{1,\dots,N+1\}$ and hence, by repeating this reasoning the system trajectory results in a sequence of states within the $DTMC$, i.e., $q_{j_0}q_{j_1}q_{j_2}\dots$. Let us assume that there exists a pair (x_i, x_{i+1}) such that there is no transition within the $DTMC$ for the corresponding pair $(q_{j_i}, q_{j_{i+1}})$. However, this contradicts the fact that by construction of the abstraction, if $x_{i+1} \in ReachableSet(\mathcal{P}_{j_i})$ there exists a transition from q_{j_i} to $q_{j_{i+1}}$ with a certain probability. Hence, for all pairs (x_i, x_{i+1}) it holds that $T_{MC}(j_i, j_{i+1}) \neq 0$, and thus, the abstraction is complete. ■

Although the abstraction is complete, it introduces also some false transitions. In order to minimize the amount of false transitions, the probabilities associated to the transitions between the discrete states are validated using Monte Carlo Simulation, as specified next in Algorithm 2. Based on its output we will be able to define an *abstraction error*. Monte Carlo simulation is done as follows: for each cell $\mathcal{P}_i$, a number of state space samples are randomly generated, which results in a discretized cell $\mathbb{P}_i$ with a finite number of points. Then for each point generated in the state space we simulate for a finite sampling of the input space $\mathbb{U}_\delta$. In what follows $\mathcal{S}_1 \oplus \mathcal{S}_2 := \{x + y \,|\, x \in \mathcal{S}_1,\, y \in \mathcal{S}_2\}$ denotes the Minkowski addition of the sets $\mathcal{S}_1$ and $\mathcal{S}_2$.

Input:

- N_x - number of state space samples;
- system dynamics (2);
- T_{sim} - zero matrix of suitable dimensions.

Output:

- T_{sim} - Monte Carlo probability matrix.

Algorithm 2 Monte Carlo Simulation

```
for all i = 1,...,N do
    find dynamics l active in P_i;
    for p = 1 : N_x do
        select x_0^p ∈ P_i as initial condition;
        compute x̄_1 ← {A_l x_0^p} ⊕ B_l U_δ ⊕ {f_l};
        for j ∈ {1,...,N+1} do
            if P_j ∩ x̄_1 ≠ ∅ then
                T_sim(i,j) ← T_sim(i,j) + 1;
            end if
        end for
    end for
    normalize T_sim(i,:) such that Σ_j T_sim(i,j) = 1;
end for
return T_sim
```

Recall that the aim of this paper is to find a state space abstraction with a trade-off between complexity and accuracy. A complete abstraction is computed with Algorithm 1, while Algorithm 2 performs "random" one-step simulations, in order to detect the false transitions introduced by Algorithm 1.

Next we can define the *abstraction error*

$$\epsilon(i,j) := |T_{MC}(i,j) - T_{sim}(i,j)| \tag{3}$$

for all admissible pairs (i,j), i.e., the error between the transition weights resulted from the state space abstraction, using Algorithm 1, and the transition weights resulted from simulation; then we can check if the abstraction error given by (3) is below a certain threshold.

The results reported in the next section show that by further increasing the grid step of the state space partition the abstraction error eventually decreases. As the grid step tends to infinity, the abstraction error tends to zero as in the limit case all transitions will have probability one. In practice, the abstraction error can be used as a stop criteria for increasing the grid step and as a confidence estimator, i.e., if the average of matrix elements is below a given threshold ϵ_{sup}, the abstraction is complete and has enough accuracy.

IV. Path generation and controller synthesis

Once the state space abstraction was computed, a satisfying path with respect to the given LTL formula must be generated and a controller that drives the system along the path must be designed. Solutions to these sub-problems of Problem II.1 are presented next.

Path generation: The next step in solving the proposed problem, after abstracting the state space to a Discrete Time Markov Chain $DTMC$, is to find a trajectory over the generated transition system which satisfies the LTL formula Φ. Note that in the discrete abstraction used in Algorithm 1 we do not use any interest regions. We denote by π_i, $i = 1, 2, \ldots, \mu$, μ predicates which are polyhedral sets. In order to express in a correct way the information about the predicates, due to the fact that the partition is not proposition preserving, the definition of the $DTMC$ is enhanced with the following function labeling description: $L : S \rightarrow 2^{AP} \times 2^{[0,1]}$, where $[0, 1]$ is the probability interval.

Thus the transitions between states have some probabilities, but also each atomic proposition associated with each state has some probability of existence. This is due to the fact that the partition is not proposition preserving; for example, if a cell $\mathcal{P}_i$ within the partition contains two propositions ϕ_1 and ϕ_2, covering 20%, respectively 80% from the cell, that means that $L(\mathcal{P}_i) = \{(\phi_1, 0.2), (\phi_2, 0.8)\}$.

For finding a path in the $DTMC$ we will construct the probabilistic product automaton between the $DTMC$ and the $Büchi$ Automaton, corresponding to the LTL formula. The structure of a $Büchi$ Automaton is given in Definition IV.1.

Definition IV.1 A $Büchi$ Automaton is a tuple $B = (S_B, S_{0_B}, \Sigma_B, \rightarrow_B, F)$ where:

- S_B is a finite number of states;
- $S_{0_B} \subset S$ is a set of initial states;
- Σ_B is the input alphabet;
- $\rightarrow_B \subseteq S \times \Sigma_B \times S$ is a nondeterministic transition relation;
- $F \subseteq S$ is a set of accepting states.

We will use an adaptation of the definition from [18] to define the probabilistic product automaton, in which we will also introduce the information containing the confidence of the state space partition.

Definition IV.2 A probabilistic product automaton $PA = DTMC \times B$ between the $Discrete\ Time\ Markov\ Chain$ and the $Büchi\ Automaton$ is a tuple $PA = (S_{PA}, S_{0_{PA}}, \rightarrow_{PA}, T_{PA}, F_{PA})$ where:

- $S_{PA} = S \times S_B$ is the set of states;
- $S_{0_{PA}} = S_0 \times S_B$ is the set of initial states;
- $\rightarrow_{PA} \subseteq S_{PA} \times S_{PA}$ is the transition relation, where $((s_i, s_{B_m}), (s_j, s_{B_n})) \in \rightarrow_{PA}$ if and only if $P(s_i, s_j) := T_{MC}(i, j) > 0$ and $(s_{B_m}, L(s_i), s_{B_m}) \in \rightarrow_B$;
- $F_{PA} = S \times F$ is a set of final states;
- $T_{PA} : S_{PA} \times S_{PA} \rightarrow [0, \infty)$ is the transition cost.

Above, the transition cost T_{PA} is defined as follows:

$$\begin{aligned} &T_{PA}((s_i, s_{B_m}), (s_j, s_{B_n})) = \\ &= -log(P(s_i, s_j)) - log \sum_{\pi \in AP, (s_{B_m}, \pi, s_{B_n}) \in \rightarrow_B} (L(s_i)) \\ &+ \epsilon(i, j), \end{aligned} \tag{4}$$

where $P(s_i, s_j) > 0$, $L(s_i) = (\pi_i, p)$ and $p \in [0, 1]$.

The transition cost uses the logarithm of the probabilities because we want to find a path with a minimum cost, which maximizes the probability of existence of the observable from the LTL formula and the probability of the transitions which can lead to a final state. The error matrix is used in the transition cost, because we want to choose the transitions with the most accurate probability. The product automaton contains the observables and transitions from $DTMC$ and the transitions from the $Büchi\ Automaton\ B$. So if an accepted path in PA is found (at least one final state is reached), this means that the LTL formula is satisfied with a certain probability (given by the cost), and there are controllers which enable transitions to each state from the resulting path (also with some probability). A path in *PA* and its projection into $DTMC$ which satisfy the LTL formula is found using tools from [6]. After all satisfying paths are found, the one with minimum cost is chosen; this means that the chosen path has the maximum probability of satisfying the formula. Naturally, other optimality criteria can also be used for selecting a path.

Controller synthesis: Next we present the controller which represents the last step in solving Problem II.1. Recall that the aim is to maximize the probability of satisfying the LTL formula. First, a satisfying path starting with the current cell is generated; then the first cell within the path is selected. Using Linear Programming, we try to find a control action $u(t) \in \mathbb{U}$ for the current state $x(t)$ such that a transition from the current cell to the intersection between the next cell in the path and its corresponding interest region is enabled. If an admissible $u(t)$ is found, the next state is computed and, the first element within the path is removed and the algorithm continues with the next cell from the path. A path is defined as $Path := (q_{j_0} q_{j_1} \ldots)$ and $Path(0) := q_{j_0}$, $Path(1) := q_{j_1}$ and so on.

If no feasible input is found the current path is discarded, the current cell becomes the initial cell and a new path is searched for within the PA. This is a possible recovery solution for controller synthesis; another option is to possibly violate the formula for one discrete time step and move to one of the other cells, where transitions are possible from the current cell, or to another state within the same cell, and then regenerate the path. A transition to another cell or to another state within the same cell will always be possible, because the sum of probabilities over all transitions leaving any of the cells is equal to one.

To compute the control action $u(t) \in \mathbb{U}$, we apply an adaptation of the polytope-to-polytope control method from

Algorithm 3 Controller Synthesis

1: $t = 0$;
2: Let $x(t) \in \mathcal{P}_{j_t}$; $Path(t) \leftarrow q_{j_t}$;
3: Generate $Path := (q_{j_t} q_{j_{t+1}} \ldots)$;
4: **while** $Path \neq \emptyset$ **do**
5: find dynamics l_t active in $Path(t)$;
6: find $u(t)$ by solving LP (5);
7: **if** LP feasible **then**
8: compute $x(t+1)$;
9: $t \leftarrow t+1$;
10: Go to 5;
11: **else**
12: $Path \leftarrow \emptyset$;
13: $x(0) \leftarrow x(t)$;
14: **end if**
15: **end while**
16: Go to 1;

[11]. Let $x(t)$ be the current state, (A_l, B_l, f_l) the corresponding dynamics, $\mathcal{P}_{j_{t+1}}$ the next cell in the path and $u(t)$ a corresponding control action. For the current state, we can define the following set of linear equalities and inequalities constraints in the variable $u(t)$:

$$\begin{aligned} x(t+1) &= A_l x(t) + B_l u(t) + f_l, \\ H_{\mathbb{U}} u(t) &\leq h_{\mathbb{U}}, \\ H_P x(t+1) &\leq h_P, \end{aligned} \tag{5}$$

where matrices $H_{\mathbb{U}}, h_{\mathbb{U}}$ and H_P, h_P are corresponding coefficients for specifying input constraints and state constraints (the next state must be in the next cell in order to enable the found path), respectively. A solution to the problem described by (5) is found by solving the corresponding Linear Programming problem.

V. Illustrative Example: Fuel Balancing

The application considered consists of two airplane fuel tanks T_1 and T_2, as originally described in [9]. The state variables are the two fuel volumes v_1 and v_2 in these tanks. The maximum capacity in both tanks is 10 fuel units. The system can be controlled by moving fuel from tank T_1 to tank T_2 by using a pump, at a maximum rate of 3 fuel units per time step. The considered system has two dynamical modes, one for normal operation and one for refueling mode. During aerial refueling mode, a tank plane is flying next to the plane which needs fuel and fills up fuel in tank T_1 at a rate of 3 fuel units per time step. Also, it is assumed that the aircraft consumes 1 fuel unit per time step from tank T_2 [9]. So this system can be described by the following model:

$$\begin{bmatrix} v_1(t+1) \\ v_2(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} u_c(t) + \begin{bmatrix} u_{in}(t) \\ -1 \end{bmatrix} \tag{6}$$

$$u_{in}(t) = \begin{cases} 3, \text{ if } v_1(t) + v_2(t) \leq 2 \text{ and } u_{in}(t) = 0; \\ 0, \text{ if } v_2(t) \geq 8 \text{ and } u_{in}(t) = 3, \end{cases} \tag{7}$$

where $u_{in}(t)$ can be either 0 or 3, depending on the dynamical mode and $u_c(t)$ is the controlled input. The system dynamics equation can be written in the form (2) as follows:

$$\sigma(t) \in \{1, 2\};$$

$$A_1 = A_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, B_1 = B_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix},$$

$$f_1 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, f_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

$$\sigma(t) = \begin{cases} 1, v_1(t) + v_2(t) \leq 2, u_{in}(t) = 0; \\ 2, v_2(t) \geq 8, u_{in}(t) = 3. \end{cases} \tag{8}$$

The constraint on the input signal is formulated as follows:

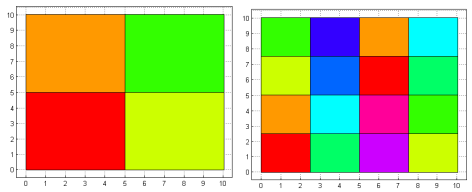
$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} u_c(t) \\ v_1(t) \end{bmatrix} \leq \begin{bmatrix} 3 \\ 0 \\ u_{in}(t) \end{bmatrix}. \tag{9}$$

In order to avoid instability of the airplane, we will control u_c such that the difference between the two fuel levels will remain low. The system must satisfy two specifications:

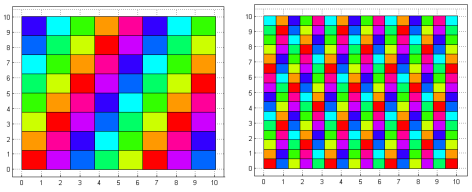
- always $|v_1 - v_2| \leq 2$;
- always eventually $|v_1 - v_2| \leq 1$.

The first specification is safety relevant and the second one is performance relevant. Then, the atomic propositions are $\pi_1 : |v_1 - v_2| \leq 2$ and $\pi_2 : |v_1 - v_2| \leq 1$ and the LTL formula which must be satisfied is $\phi = \Box\pi_1 \,\&\, \Box\Diamond\pi_2$.

1) Abstraction Algorithm 1 - Results: In Fig. 1 we plot the state space partition using different Grid Steps (1, 2, 3, 4).



(a) Grid Step = 1 (b) Grid Step = 2



(c) Grid Step = 3 (d) Grid Step = 4

Fig. 1: State Space partition - Algorithm 1.

In the table below, Step values represent in how many grid steps the State Space is partitioned. For $Step = 1$, the

x, y, z coordinates are partitioned in half. At the next step, the coordinates of all resulting cells from the previous step are partitioned again in half and so on. Time values represent the number of seconds used for constructing the partition of the state space and computing the transitions among the partitions; Nr. States is the number of states of the Markov Chain obtained using Algorithm 1. In the Monte Carlo Simulation column there is the average error between the probability on transitions given by Algorithm 1 and the probability on transitions given by Algorithm 2 for both dynamical modes. The result from the Confidence Flag column is computed using Algorithm 2 and it states whether a certain partition has the error between the transitions computed using Algorithm 1 and the simulations below a given threshold or not. M1 and M2 represent the two dynamical modes of the system - normal and aerial refueling.

After running Algorithm 1 using a grid step 1, 2, 3 and 4, respectively, and using a threshold error = 0.007, for which the confidence of the results are considered correct compared with Monte Carlo Simulation, we obtained the following result:

	Markov Chain		Monte Carlo		
Step	Time	Nr. States	M1	M2	Confidence
1	0.11	4	0.026	0.045	0
2	0.17	16	0.014	0.020	0
3	0.23	64	0.004	0.006	1
4	0.76	256	0.0012	0.0019	1

TABLE I: Confidence results

The obtained results indeed validate the fact the the proposed abstraction method allows for a trade-off between accuracy and complexity and manages to yield a sufficiently accurate abstraction with a significantly reduced number of regions (i.e., 64) compared with the approach of [9] (277 regions).

2) Controller Synthesis: In this subsection we present the results after applying the control synthesis procedure of Algorithm 3. For each partition - GridStep = 1, 2, 3, 4 - the simulations are done and the results are presented in Fig. 2, Fig. 3, Fig. 4, Fig. 5 respectively.

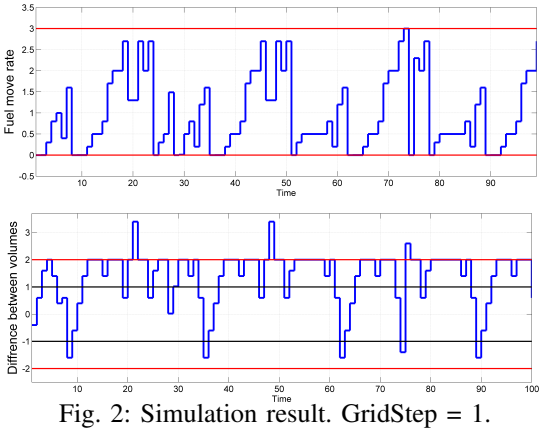


Fig. 2: Simulation result. GridStep = 1.

The first graphic (Fuel move rate) represents the time evolution of input u_c and its polytopic constraint: $0 \leq u_c \leq 3$. The second graphic (Difference between volumes) represents the time evolution of the satisfaction of the LTL formula. The red line represents the safety requirement $|v_1 - v_2| \leq 2$ and the black line represents the performance requirement $|v_1 - v_2| \leq 1$.

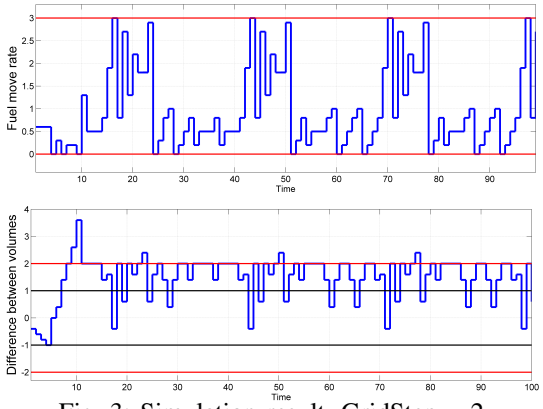


Fig. 3: Simulation result. GridStep = 2.

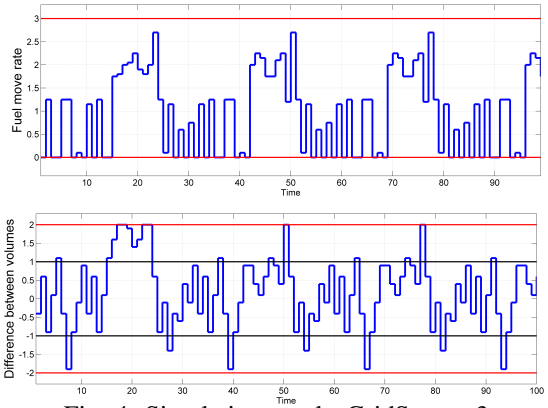


Fig. 4: Simulation result. GridStep = 3.

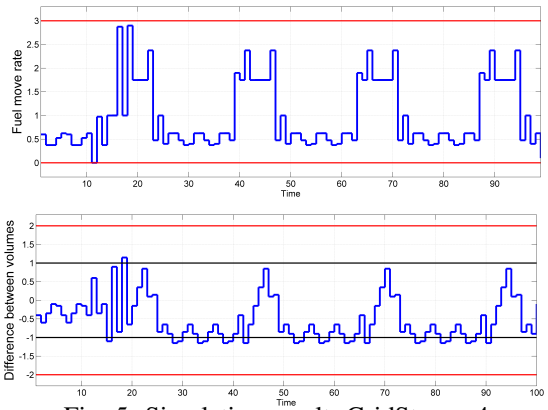


Fig. 5: Simulation result. GridStep = 4.

From the verification done with Monte Carlo Simulation we can see that only the partitions with grid step 3 and 4 are

reliable. Hence, we can observe in Fig. 2 and Fig. 3 which correspond to grid step 1 and 2, respectively, that even the safety specification is violated. In Fig. 4 which corresponds to grid step 3, the safety requirement is satisfied always and the performance requirement is often met; in Fig. 5 the safety requirement is always satisfied and the system performance is high.

3) Analysis of the results: Using the algorithm from [9], the complexity of the resulting partition can not be specified a priori as in Algorithm 1, where the complexity of the partition can be prespecified. The algorithm used in [9] produces a result that satisfies the specification; Algorithm 1 produces also a result that satisfies the LTL formula but only if the complexity specified a priori gives an accurate enough partition. Verification is done comparing the Algorithm 1 output with Monte Carlo Simulation. In conclusion, the specification can be fulfilled using Algorithm 1, with a prespecified complexity with lower number of states and with the resulting partition verified. The comparison results are summarized in the table below:

Grid Step	Predefined complexity	No. states	Obs. Formula Satisfaction
1	✓	4	✗
2	✓	16	✗
3	✓	64	✓
4	✓	256	✓

TABLE II: Algorithm 1. Results

These results confirm the benefits of the proposed approach: explicitly trading off complexity versus accuracy, as done by Algorithm 1, can result in attaining the same performance with less complexity.

VI. Conclusions

In this paper we have proposed a probabilistic approach to the state space abstraction problem for deterministic systems and temporal logic control. The main contribution was to develop an abstraction model and algorithm that allow for a priori specifying the desired complexity for the resulting abstraction. As such, the proposed approach allows a trade-off between the accuracy and complexity of the resulting abstraction. As future research we are interested in improving scalability of the abstraction algorithm by assigning transitions without computing reachable sets.

Acknowledgements

The first author gratefully acknowledge financial support by the German Research Foundation (DFG) Graduiertenkolleg 1480 (PUMA) and the Erasmus Mobility Grant.

References

[1] E. A. Emerson, *Temporal and Modal Logic*, J. van Leeuwen, Ed. Amsterdam: Elsevier, 1990.

[2] M. Antoniotti and B. Mishra, "Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion controllers," in *IEEE International Conference on Robotics and Automation*, 1995, pp. 1441 – 1446.

[3] C. Kloetzer, M.and Belta, "LTL Planning for Groups of Robots," in *IEEE International Conference on Networking, Sensing and Control*, 2006, pp. 578–583.

[4] G. Batt, D. Ropers, H. De Jong, J. Geiselmann, R. Mateescu, and D. Schneider, "Validation of qualitative models of genetic regulatory networks by model checking: Analysis of the nutritional stress response in Escherichia coli," *Bioinformatics*, vol. 21, pp. 19–28, 2005.

[5] G. Batt, C. Belta, and R. Weiss, "Temporal Logic Analysis of Gene Networks under Parameter Uncertainty," *IEEE Transactions of Automatic Control*, vol. 53, pp. 215–229, 2008.

[6] M. Kloetzer and C. Belta, "A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications." *IEEE Trans. Automat. Contr.*, vol. 53, pp. 287–297, 2008.

[7] P. Tabuada and G. J. Pappas, "Model Checking LTL over Controllable Linear Systems Is Decidable." in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, vol. 2623. Springer, 2003, pp. 498–513.

[8] P. Gastin and D. Oddoux, "Fast LTL to Büchi Automata Translation." in *Proceedings of the 13th International Conference on Computer Aided Verification*, ser. CAV '01. London, UK: Springer-Verlag, 2001, pp. 53–65.

[9] P. Nilsson, N. Ozay, U. Topcu, and R. M. Murray, "Temporal logic control of switched affine systems with an application in fuel balancing," in *American Control Conference, ACC*, 2012, pp. 5302–5309.

[10] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLiP: a software toolbox for receding horizon temporal logic planning," in *Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011*, 2011, pp. 313–314.

[11] E. A. Gol, M. Lazar, and C. Belta, "Language-guided controller synthesis for discrete-time linear systems," in *Hybrid Systems: Computation and Control (part of CPS Week 2012), HSCC'12, Beijing, China, April 17-19, 2012*, 2012, pp. 95–104.

[12] A. Abate, A. Ames, and S. Sastry, "Stochastic Approximations of Hybrid Systems," in *Proceedings of the 24th American Control Conference*, 2005, pp. 1557–1562.

[13] M. Althoff, O. Stursberg, and M. Buss, "Model-Based Probabilistic Collision Detection in Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[14] J. Lunze and B. Nixdorf, "Representation of Hybrid Systems by Means of Stochastic Automata," *Mathematical and Computer Modelling of Dynamical Systems: Methods, Tools and Applications in Engineering and Related Sciences*, vol. 7, no. 4, pp. 383–422, 2001.

[15] A. Abate, J. Katoen, J. Lygeros, and M. Prandini, "Approximate Model Checking of Stochastic Hybrid Systems," *European Journal of Control*, vol. 16, no. 6, pp. 624–641, 2010.

[16] S. E. Z. Soudjani and A. Abate, "Adaptive Gridding for Abstraction and Verification of Stochastic Hybrid Systems," in *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems*, 2011, pp. 59–69.

[17] A. D'Innocenzo, A. Abate, M. D. Benedetto, and S. Sastry, "Approximate Abstractions of Discrete-Time Controlled Stochastic Hybrid Systems," in *Proceedings of the 47th IEEE Conference of Decision and Control*, 2008, pp. 221–226.

[18] N. Ghita, M. Kloetzer, and O. Pastravanu, "Probabilistic car-like robot path planning from temporal logic specifications," *15th International Conference on System Theory, Control, and Computing (ICSTCC)*, pp. 1–6, 2011.