Research Plan

Synopsis of The Proposed Research Plan

Submitted for the Degree of

Doctor of Philosophy in

Department of Computer Science and Engineering

National Institute of Technology Meghalaya

By

Amit Gurung

Roll No: ***P14CS002***

Under the Supervision of

Dr. Rajarshi Ray

HOD, Computer Science and Engineering,
National Institute of Technology Meghalaya

Title of research proposal : Parallelizing Reachability Analysis of Hybrid System.

Broad Research Area : Formal methods for safety verification of Hybrid Systems.

Specific Research Area : Reachability Analysis of Hybrid Systems.

Name of the Research Scholar : Amit Gurung

Name & Designation of Supervisor : Dr. Rajarshi Ray

HOD, Computer Science & Engineering

National Institute of Technology Meghalaya

(Dr. Rajarshi Ray)
Supervisor

(Amit Gurung)
Research Scholar

Contents

1 Introduction

Our life are becoming more and more dependent on technology and products produced out of technological development. The use of mobile phones, tablets, computers and laptops, software services like the email and social networking sites, online bank transactions, online shopping sites such as flipkart, amazon, ebay, e.t.c., and many more have become our daily needs. We can sense that this human dependence on technology is going to increase in the future. These dependence on technology compels us to establish their correctness or perfectness. Imperfections can only be tolerated to some extent for applications which are not so critical but not otherwise. Eg., we can tolerate if the text editor fails to retain the format last saved or an operating system in our laptop crash when an audio player is run or an email sent to one person ends up in the inbox of someone else, but we cannot tolerate slightest error in, for example, the Traffic Alert and Collision Avoidance System for air traffic(or train traffic) control which might lead to a collision. In fact, we are becoming less and less tolerant regarding errors in technology.

For example, a large number of lives and property loss or damage has already been incurred in airplane accidents (for instance Air France Flight 447: 12 crew members (3 flight crew, 9 cabin crew) and 216 passengers, from thirty-two nationalities on board were killed), due to hardware and software design error as reported in (`http://www.bea.aero/en/enquetes/flight.af.447/flight.af.447.php`). The details of the accident is elaborated in detailed in wikipedia titled "Air France Flight 447". Thus, safety verification of such critical system is an utmost concern. Such incidents urges the importance of design specification and design validation before the deployment of such critical system.

To overcome these technological imperfection in critical systems **formal methods** is a must. *Formal methods are mathematical techniques for the specification, development and verification of software and hardware systems*. The goal of formal methods is to contribute to the reliability and robustness of a software or hardware system. The application of formal methods in real world is increasing and with the recent development of powerful tools which are scalable, the future looks brighter for formal methods based techniques in design validation.

Simulation, testing and deductive verification are traditional approaches to gain greater confidence on systems[1]. While simulation is carried out on the model of a design, testing is performed on the actual design itself. Deductive reasoning is a mathematical proof system where correctness of systems are proved with axioms and proof rules. We know that simulation and testing are inadequate in establishing total confidence of the design under validation because they are not exhaustive checks. Deductive verification can be extremely expensive at times

Model checking is an automatic technique for verifying finite state systems. The procedure normally uses an exhaustive search of the state space of the system to determine if a specification is true or not. There are broadly two types of system properties those are checked with model checking algorithms, namely **safety** properties and **liveness** properties[1].

Safety properties are properties which specify that nothing bad occurs.

Liveness properties are properties which specify that something good eventually occurs.

There are model checking algorithms which are reasonably efficient and allows for its automation. Infinite state systems can also be model checked with abstractions which constructs finite symbolic states from the infinite state space.

Model checking, however, suffers from the ***state space explosion problem*** [2, 3, 4, 5]which arises due to the exponential increase in the number of explicit states of a system. State space explosion problem can be tackled to some extent with ***model abstractions, use of efficient data structures, heuristics and symbolic representation of states***[1].

As mentioned in [1], applying model checking for **design validation** mainly involves three steps: (1) Modeling, (2) Specification and (3) Verification. *Modeling* is the process of formalizing a design with a mathematical model. A model is sometimes abstracted to hide unnecessary details and to make

it within analysis limits and trying to cope with the state space explosion problem. *Specification* means formally stating the properties that the design must satisfy. *Verification* is the process of automatically checking if the given specifications are satisfied by the design under validation. If a specification is found to be violated, a counter example is expected to be returned by the model checking algorithm showing the design behavior that violated the given specification.

Some examples of model checkers available today are SPIN[6], UPPAAL[7], Kronos[8], HyTech[9], PHAVer[2] and SpaceEx[10]. SPIN is a model checker for distributed software systems against Linear Temporal Logic (LTL) specifications. UPPAAL is a model checker for real time systems modeled with timed automata. Kronos is similarly a model checker for real time systems modeled with timed automata against TCTL (Timed Computation Tree Logic) specifications. HyTech and PHAVer are model checkers for hybrid systems modeled as linear hybrid automata. SpaceEx, a new extensible verification platform for hybrid systems, developed with systematic software engineering, and featuring a web-based graphical user interface.

1.1 Hybrid Systems

Hybrid systems is a combination of two complementary units - the computational unit and the physical unit. The computational unit is the discrete system where as the physical unit is the continuous system. This systems currently referred to as cyber-physical-systems (CPS) [11].

Though the emphasis tends to be more on the computational units, but the hybrid systems works as a feed-back loop between the computational and physical units and vice-versa. The computational unit in a hybrid systems usually drives the physical system.

The Hybrid systems is also referred to as embedded systems, it has a large area of applications, where the physical system is generally controlled by the computational system in a feed-back loop. Some of the day-to-day applications are refrigeration in supermarkets, air conditioning systems in buildings, automatic gear controlling systems in automobiles, aerospace, chemical processes, etc.

1.2 Hybrid Automata

Hybrid systems are modelled by hybrid automaton. A hybrid automaton as stated in[12] is an finite state machine which consists of a number of state representing a mode or finite sets of real-valued variables comprising of discrete states. These state or modes are linked by annotation (labeling) with constraints that specify the continuous evolution of the system, also the link between the modes are annotated with guards (or barriers) that determine the discrete transitions condition. The discrete transitions or jumps changes the continuous dynamics and modify the values of the continuous variables. The jumps only take place when the values of the variables attains a certain limit, called guard.

The hybrid automaton can be mapped to a graph, where states are vertices or nodes of the graph and linked annotations are the edges of the graph. The system may only remain in a particular state as long as the variable values are in a range called invariant associated with the state.

Formally, a hybrid automaton $H = (Loc, Var, Lab, Trans, Flow, Inv, Init)$ consists of the following elements:

• vertices, called locations, are given by a finite set Loc, and whose edges, called discrete transitions, are given by a finite set Trans;

• a finite set of real-valued variables Var . A state of the automaton consists of a location and a value for each variable (formally described as a valuation over Var). The set of all states of the automaton is called its state space. To simplify the presentation, we assume that the state space is $Loc \times \mathbb{R}^n$, where n is the number of variables. We will also simply write x to denote the name of the variable x or its value according to the context;

• for each location, the variables can only take values in a given set called *invariant*. The invariant is given by $Inv \subseteq Loc \times \mathbb{R}^n$;

• for each location, the change of the variables over time is defined by its time-derivative that must be in a given set $Flow \subseteq Loc \times \mathbb{R}^n \times \mathbb{R}^n$. For example, if the system is in a location l , a variable x can take the values of a function $\xi(t)$ if at each time instant t , $(l, \dot{\xi}(t), \xi(t)) \in Flow$, where $\dot{\xi}(t)$ denotes the derivative of $\xi(t)$ with respect to time;

• the discrete transitions $Trans \subseteq Loc \times Lab \times 2^{\mathbb{R}\times\mathbb{R}} \times Loc$ specify instantaneous changes of the state of the automaton. A transition $\left(l, \alpha, \mu, l'\right)$ signifies the system can instantaneously jump from any state$(l, x)$to any state $\left(l', x'\right)$ if $x' \in Inv(l')$ and $\left(x, x'\right) \in \mu$. Every transition has a synchronization $\alpha \in Lab$ that is used to model the interaction between several composed automata. Intuitively, if two automata share a common α, transitions with this can only be executed in unison, i.e., by simultaneous execution of a transition with this label in both automata. The relation μ is called the jump relation of the transition;

• A set of states $Init \subseteq Loc \times \mathbb{R}^n$ specifies the initial states from which all behavior of the automaton begins.

1.3 Reachability Analysis of Hybrid Systems

Safety verification is the verification of functional correctness of the system. Safety verification is the process of finding if any trajectories emerging from the given initial configuration of a mathematical model of the system passes or intersects an unsafe or bad states of that system. So, safety verification includes the process of computing all the possible behaviours or states that hybrid systems may pass through and checking if there exists an unsafe/unwanted behaviour or state.

This technique of computing and checking exhaustively all the possible states of the system or model is used in the model checking algorithms[3].

An algorithmic approach[13], based on the computation of the reachable set, has emerged from hybrid systems research. A reachable set consists of all the states that can be visited by a trajectory of the hybrid systems starting from a specified set of *initial states*. A *trajectory[1]* is the path constituting all the states starting from the initial state that the system can take under its dynamics.

If the size of the initial state is infinite and the trajectories are non-deterministic, then the computation of all the reachable set is a challenge. Hence, there is a great scope for research in this directions.

Methods developed so far in these area for computation of reachable set are limited to small time bound as the amount of time taken to compute all the reachable states of any hybrid systems is hard. A number of approaches are going on in order to increase the process of computing reachability set as mentioned in the paper [13].

One of the well-know model checking tool available today, SpaceEx (http://spaceex.imag.fr)[10] is consider to take much less time then many others, but even such a tool chokes for high dimensional hybrid systems such as a helicopter controller model when supplied with a fairly large time bound for the computation of its reachable states.

2 Review of literature

2.1 Reachability Analysis

Reachability analysis concerns the computation of the system's reach set, the set of reachable states in the state space from a given set of initial states. A trajectory[1] of a dynamical system can be computed with numerical simulations for a given initial state and input using numerical integration. Simulation is efficient for design validation to a certain limit. There are simulators available for dynamical systems such as the MATLAB Simulink for model based development of complex systems, simulations can also provide the designers with an overall idea of the reachable set by choosing some wise start points like the boundary values of the initial set. However, simulations in general cannot guarantee safety or liveness properties. Reachable set on the other hand if computed can guarantee safety.

A hybrid automaton consists of potentially infinite number of states. Algorithmic analysis need a finite representation of the infinite state space and that is done through symbolic state representation. A symbolic state is a pair of a discrete set and a continuous set. The discrete set is a set of state locations and the continuous set gives the value of the continuous variables of the hybrid systems in that location(s).

Reachability computation is a process of searching exhaustively all symbolic states till the fix-point has reached and if the fix-point does not exist, then bounded reachability can be computed which is computing all reachable states within a fix time bound T.

$$R(T) = \{x \, \epsilon \, \mathbb{R}^n | \exists x_0 \epsilon I, \; t \epsilon [0, T] \; such \; that \; \Pi x_0(t) = x\}$$

where I is the initial set and $\Pi x_0(t)$ denotes the state of the trajectory initiated from x_0 at time t.

We, are also interested in the states of the system between a time interval which is defined as follows:

$$R(t_1, t_2) = \{x \, \epsilon \, \mathbb{R}^n | \exists x_0 \epsilon I, \; t \epsilon [t_1, t_2] \; such \; that \; \Pi x_0(t) = x\}$$

The reachability problem of a hybrid automaton H is to compute all the reachable set of a trajectory which begins from the initial value x_0 and ends in x_f . In most of the cases the reachable set is infeasible to compute and the exact reachablilty analysis is an undecidable problem. So, researchers compute this undecidable system by implementing an over-approximation technique[4, 14, 15] to compute the reachable set. As the over-approximated reachable set will include all the reachable set plus some finite extra set so as to be sure that the trajectory will not exclude the actual reachable path. It is then checked if this over-approximated set is safe, i.e., it does not intersect with the bad set. An over-approximated set Z_{over} contains more states than the model actually reaches, i.e., $R \subseteq Z_{over}$. Safety of the over-approximated set implies the safety of the design under validation, as it overly include all the reachable states plus some extra states which may not be reachable. If there is an intersection of the over-approximated set with the bad set say F, then the system is unsafe, which may not be true in all cases in actual. Mathematically, if $Z_{over} \cap F = \emptyset$ then that system is safe. The process of over-approximating certainly helps in two aspects. Firstly, it makes sure that no bad states of the reachable sets are left in computation which guarantees the proper verification of any system's safety. Secondly, it helps in converting the problem of computation of all actual reachable sets from infeasible to feasible.

There have been a number of methods undertaken in order to compute reachablity set as described in the papers [1, 16, 17, 18, 19, 20, 15]. Some of these methods are described in brief in the following sections.

2.2 Existing work on computing reachability Set

2.2.1 Support Functions

Defining Support Functions

The support function can be computed for any convex set as described in [15, 14]. Support function can be used to compute the reachability set and in turn be used for reachability analysis of a hybrid systems. The support function of any compact convex set $\Omega \subseteq R^d$, denoted by ρ_Ω, is defined as :

$\rho_\Omega : R^d \rightarrow R$

$l \mapsto max_{x\epsilon\Omega}\ (l \cdot x)$

Also, the support vector of a compact convex set $\Omega \subseteq R^d$, in the direction of $l \subseteq R^d$, is denoted by $v_{\Omega,l}$, is a vector of R^d such that:

$v_{\Omega,l}\epsilon\,\Omega$ and $l \cdot v_{\Omega,l} = \rho_\Omega(l)$.

Generally, the support vector of Ω in any direction l is not always unique, as if we take the normal perpendicular to the direction l, then all the points on the normal/line perpendicular to the direction l is the result of the support function. It is to be noted that a compact convex set Ω is uniquely determined by its support function as the following equality holds:

$$\Omega = \cap_{\ell\epsilon R^d} x\epsilon R^d : l \cdot x \leq \rho_\Omega(l). \tag{1}$$

Where $\ell \in R^d$ and $\ell_1, \ell_2, ..., \ell_r$ is uniformly distributed through out the convex set. From equation (1), it is easy to see that a tight polyhedral over-approximation of an arbitrary compact convex set can be obtained by "sampling" its support function uniformly distributed over ℓ as described in [15, 14].

If Ω is a compact convex set and $\ell_1, \ell_2, ..., \ell_r$ be arbitrary chosen vectors, then the halfspaces H_i, is denoted as

$$H_i = x\epsilon R^d : \ell_i \cdot x \leq \rho_\Omega(\ell_i), \quad i = 1, ..., r. \tag{2}$$

Thus, a polyhedron can be defined by the intersection of all these halfspaces of equation (2) i.e., $\tilde{\Omega} = \cap_{i=1}^{r}\ H_i$ as stated in the paper [15, 14]. Then, $\Omega \subseteq \tilde{\Omega}$ and the process result in a tight over-approximations as Ω touches the faces of $\tilde{\Omega}$ at points $v_{\Omega,l_1}, v_{\Omega,l_2}, ..., v_{\Omega,l_r}$. Using support vectors and support function reachability of most of the classes of sets can be easily computed.

If A is any matrix and $U, V \subseteq R^d$ be any compact convex sets, and all non-zero vectors $l \in R^d$, then some of the well-known properties of support function are as follows:

$\rho_{CH(U,V)}(l) = max(\rho_U(l), \rho_V(l))$

$\rho_{U\oplus V}(l) = \rho_U(l) + \rho_V(l)$

$\rho_{AU}(l) = \rho_U(A^T l)$

where CH(U,V) denotes the convex hull of U and V.

Using these properties, unusual convex sets can also be easily consider for the computation of reachability analysis as described in the papers [15, 14].

Reachability computation of Discrete-Time systems using Support Functions

A discrete-time linear system of the form:

$$x_{k+1} = Ax_k + Bv_k, \quad x_0\epsilon I, \quad v_k\epsilon V \tag{3}$$

where $I \subseteq R^d$ and $V \subseteq R^d$ are compact convex sets. Now if we denote the convex sets by Ω_k the subset of states reachable at time k. The reachable sets of a hybrid system within a limited time bound N, is the sequence of sets $\Omega_1, \Omega_2, ..., \Omega_N$ as described in [14]. Thus, the recurrence relation can be computed as shown in equation (4) below:

$$\Omega_{k+1} = A\Omega_k \oplus V, \quad \Omega_0 = I \tag{4}$$

Finally, for computing all the sequences of over-approximated reachable sets Ω_k, for $k = 0, ..., N$, with arbitrary chosen vectors $l_i \in R^d$, for all $i = 1, ..., r$ is given by the equation (5) as described in [14] is:

$$\rho_{\Omega_k}(l) = \rho_I((A^T)^k l) + \sum_{i=0}^{k-1} \rho_V((A^T)^i l) \tag{5}$$

Using the equation (5) the reachability sets can be computed as shown in the Algorithm 1. The Algorithm 1 computes the support functions $\rho_{\Omega_0}, \ldots, \rho_{\Omega_N}$.

Algorithm 1 Computation of support functions $\rho_{\Omega_0}, \ldots, \rho_{\Omega_N}$.

Input: The matrix A, the support functions ρI and ρV , the vector l and an integer N.
Output: Y_k=ρ_{Ω_k} (l) for k in $0, ..., N$

$r_0 \leftarrow l$

$S_0 \leftarrow 0$

$Y_0 \leftarrow \rho_I(r_0)$

for k from 0 to $N-1$ do

$r_{k+1} \leftarrow A^T\ r_k$
$s_{k+1} \leftarrow s_k + \rho_V\ (s_k)$
$Y_{k+1} \leftarrow \rho_I(r_{k+1}) + s_{k+1}$

end for

return $Y_0, ..., Y_N$

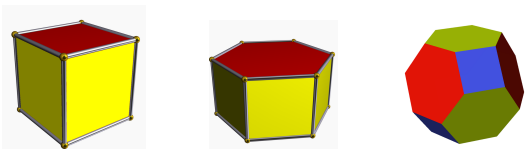
2.2.2 Zonotopes

Defining Zonotopes

Zonotopes are a special class of convex polytopes. A zonotope is a Minkowski sum of a finite set of line segments. As defined in [16], a zonotope Z is a set such that:

$$Z = \left\{ x \in R^n : x = c + \sum_{i=1}^{i=p} x_i g_i, -1 \leq x_i \leq 1 \right\} \tag{6}$$

where $c,\ g_1, ..., g_p$ are vectors of R^n. We note $Z = (c, < g_1, ..., g_p >)$. Thus, a zonotope is a polytope.



(a) Cube (b) Hexagonal Prism (c) Truncated Octahedron

Figure 1: Zonohedrons with 3,4 and 6 generators respectively.

A zonotope $Z = (c, < g_1, ..., g_p >)$ is always centrally symmetric and that the point $c \in R^n$ is the center of Z. The collection of vectors $g_1, ..., g_p$ is called the set of generators of Z. The figure (1) an extract from wikipedia titled "Zonohedron" shows a list of zonotopes and the number of generators for each of them. For a zonotope with p generators in R^n, the value of p/n is called the order of the zonotope. For instance, a parallelepiped is a zonotope of order 1. The equation (6) gives an efficient representation of the set since the number of faces of a zonotope in R^n with p generators is in $O(p^{n-1})$. In the paper [16] states that there are two important properties that motivates the use of zonotopes for over-approximating the reachable sets of any uncertain linear systems. They are as below:

1. *Zonotopes are closed under linear transformation*. Let L be a linear map and $Z = (c, < g_1, ..., g_p >)$ a zonotope,

$$LZ = \left\{ Lx \in R^n : x = c + \sum_{i=1}^{i=p} x_i g_i, -1 \leq x_i \leq 1 \right\} \tag{7}$$

   Thus, $LZ = (Lc, < Lg_1, ..., Lg_p >)$. The image of a zonotope by a linear map can be computed in linear time with regard to the order of the zonotope.

2. *Zonotopes are closed under Minkowski sum*. If $Z_1 = (c_1, < g_1, ..., g_p >)$ and $Z_2 = (c_2, < h_1, ..., h_q >)$ are two zonotopes, then,

$$Z_1 + Z_2 = (c_1 + c_2, < g_1, ..., g_p, h_1, ..., h_q >). \tag{8}$$

   Thus, the Minkowski sum of two zonotopes can be computed by the concatenation of two lists.

Approximation of Reachable Sets

The paper [16] presents the computation of the approximation of reachable sets for the following uncertain linear system given by:

$$x^{'}(t) = Ax(t) + u(t), \parallel u(t) \parallel \leq \mu \tag{9}$$

where A is an $n \times n$ matrix and $\parallel . \parallel$ denotes the infinity norm on R^n ($\parallel x \parallel = max_{i=1}^{i=n} |x_i|$). For a given set of possible initial values I, the reachable set of the system at the time t is

$$\Phi_t(I) = y \in R^n : \exists x \quad solution\ of\ (9), \quad x(0) \in I \wedge x(t) = y.$$

Thus, the reachable set R, on the interval $[\underline{t}, \overline{t}]$ from the set of initial values I can be computed as

$$R_{[\underline{t},\overline{t}]}(I) = \bigcup_{t \in [\underline{t},\overline{t}]} \Phi_t(I). \tag{10}$$

As computation of reachable set is expensive, over-approximation of the reachable set is computed using conservative approach as described in the paper [16]. The result of the equations is summarised as below:

$$R_{[0,r]}(Z) \subseteq P + \square(\alpha_r + \beta_r)$$

where $P = (\frac{c + e^{rA}c}{2}, < \frac{g_1 + e^{rA}g_1}{2}, \ldots, \frac{g_p + e^{rA}g_p}{2}, \frac{c - e^{rA}c}{2}, \frac{g_1 - e^{rA}g_1}{2}, \ldots, \frac{g_p - e^{rA}g_p}{2} >)$ is an approximated zonotopes for simplifying the computations, which is the convex hull of Z and $e^{rA}Z$, with the bloating ball of radius $\alpha_r = (e^{r\|A\|} - 1 - r\|A\|) sup_{x \in Z} \|x\|$ and $\beta_r = \frac{e^{r\|A\|} - 1}{\|A\|}\mu$, where $\square(\beta_r)$ denotes the ball of center 0 and of radius β_r for infinite norm.

2.2.3 Ellipsoids

Defining Ellipsoids

An ellipsoid is the image of an Euclidean ball by an invertible linear transformation, A. They are usually represented by their center c and a shape matrix $Q = AA^T$[20] . If it is given a point c and a *positive definite matrix* Q we can define the **ellipsoid** $E(c, Q)$ of center c and shape matrix Q as:

$$E(c, Q) = \{x : (x - c), Q^{-1}(x - c) \leq 1\} \tag{11}$$

where Q is the identity matrix I, $E(c, I)$ is the unit Euclidean ball of center c.

The Reachability Problem using Ellipsoids

Let us consider a linear equation,

$$\dot{x} = A(t)x + B(t)u,\ t_0 \leq t \leq t_1 \tag{12}$$

where $x \in R^n$is the state and $u \in R^m$ is the control as described in [20]. The matrices $A(t)$, $B(t)$ are continuous and the system is completely controllable. Moreover, the control $u = u(t)$ is any measurable function restricted by hard bounds $u(t) \in P(t)$, for almost all t, where $P(t)$ is a nondegenerate ellipsoid continuous in t, namely, $P(t) = E(q(t), Q(t))$, and

$$E(q(t), Q(t)) = \{u : (u - q(t), Q^{-1}(t)(u - q(t)) \leq 1\} \tag{13}$$

with $q(t) \in R^m$ (the center of the ellipsoid) and positive definite matrix function $Q(t) \in R^{m \times m}$ (the matrix of the ellipsoid) continuous in t. The support function of the ellipsoid is given by [20],

$$\rho(l|E(q(t), Q(t))) \quad = \quad max\{(l, x)|x \in E(q(t), Q(t))\} = (l, q(t)) + (l, Q(t)l)^{1/2} \tag{14}$$

The continuity of $Q(t)$ means that its support function $\rho(l|Q(t))$ is continuous in t uniformly in l with $(l, l) \leq 1$.

Given position $\{t_0, x^0\}$, the reach set $X(\tau, t_0, x^0)$ at time $\tau > t_0$ from this position is the set $X[\tau] = X(\tau, t_0, x^0) = \{x[\tau]\}$ of all states $x[\tau] = x(\tau, t_0, x^0)$ reachable at time τ by system (12), with $x(t_0) = x^0$, through all possible controls u that satisfy the constraint (13). The set-valued function $\tau \to X[\tau] = X(\tau, t_0, x^0)$ is known as the **reach tube** as stated in [20]. The reach set $X(\tau, t_0, X^0)$ (at time τ, from set $X^0 = X(t_0)$) is the union

$$X(\tau, t_0, X^0) = \bigcup X(\tau, t_0, x^0)|x^0 \in X^0$$

The set-valued function $\tau \to X[\tau] = X(\tau, t_0, X^0)$ is known as the **reach tube from set** X^0 .

Efforts are on to derive a single equation that would produce a sub-optimal (w.r.t. volume) ellipsoidal approximation to the exact reach set. However, it turns that ellipsoidal methods allow exact representations of the reach sets and tubes for linear systems through parameterized families of both external and internal ellipsoids, paper [20] explains the detailed approach.

2.2.4 Polytope

A polytope in R^n is a bounded intersection of a finite set of halfspaces $P = \{x|\pi_i^T x \leq d_i, x \in R^n\}$ where $\pi_i \in R^n$, $d_i \in R$ and $i = 1, \ldots, m$. Given $\Pi = [\pi_1 \ldots \pi_m]^T \in R^{m \times n}, d = [d_1 d_2 ... d_m]^T \in R^m$, we use $P(\Pi, d)$ as a short-hand notation for a polytope $P = \{x|\Pi x \leq d\} \subseteq R^n\}$.

Polytopes are closed under the following operations as mentioned in the paper [17] :

- Affine transformation: For $A \in R^{m \times n}, b \in R^m :\ AP + b = \{y|y = Ax + b, x \in P \subset R^n\} \subseteq R^m$.

- Intersection: $P_1 \cap P_2 = \{x | x \in P_1 \wedge x \in P_2, P_1 \subseteq R^n, P_2 \subseteq R^n\} \subseteq R^n$.
- Minkowski sum: $P_1 \oplus P_2 = \{y | y = x_1 + x_2, x_1 \in P_1 \subseteq R^n, x_2 \in P_2 \subseteq R^n\}$.
- Cartesian product: $P_1 \otimes P_2 = \{[x_1^T, x_2^T]^T | x_1 \in P_1 \subseteq R^n, x_2 \in P_2 \subseteq R^m\}$.

Since every $d - polytope$ in R^n is affinely equivalent to a $d - polytope$ in R^d, it is convenient to represent the $d - polytope$ in high-order space as an affine transformation of the $d - polytope$ in low-order space. For polytope $P \subseteq R^n$ satisfying $P = \Phi P_w + \gamma$ where $\Phi \in R^{n \times d}$ is full-column-rank, $\gamma \in R^n$ and P_w is a $d - polytope$ in R^d.we write $P =< \Phi, \gamma, P_w >$ and call it the affine representation for P. We enforce the following restrictions on affine representations

P_w is full-dimensional, $0 \in P_w$ and $sup_{w \in P_w} ||w|| \leq 1$.

Computation for operations on polytopes can be performed on their affine representations. For affine systems, the reach set segments $Reach([t_{k-1}, t_k])$ are generally full-dimensional, even though X_{k-1} and X_k may be low-dimensional. Therefore, low-dimensional polytopes are approximations of the actual reach sets. Over-approximations can be computed from the low-dimensional polytopes by "bloating" as described in details in the paper[17].

Reach Set Computation Using Affine Representations

For affine systems S, we consider the problem of computing $Reach([0, t_f])$ for the set of initial states $X_0 =< \Phi_0, \gamma_0, P_w >$, which is a $d - polytope$. Note that the reach set X_k at a time t_k is an affine transformation of X_0 as derived in the paper [17], which is given by:

$X_k = \varphi_0(A, t_k) X_0 + t_k \varphi_1(A, t_k) b$.

$X_k =< \varphi_0(A, t_k)\Phi_0,\ \varphi_0(A, t_k)\gamma_0 + t_k \varphi_1(A, t_k) b,\ P_w >$.

3 Issues/Research Scope in the area

Methods developed so far in the area for computation of reachable set are limited to a time bound as the amount of time taken to compute all the reachable states of any hybrid systems is infeasible. Moreover, this time bound is usually small for the same reason. A number of approaches are under going in order to scale the limit of time bound by reducing the computation time in the process of computing reachability set as mentioned in the paper [13]. In the previous section of this research proposal only some of these methodologies have been summarised.

One of the well-know system tool available today which is consider to take less time then many others with as many as 100 variables/dimension is SpaceEx (http://spaceex.imag.fr) it uses the SFA to compute reachable set as published in the paper [10]. (The tool uses the package GLPK out of many available package for solving the Linear Programming Problem of the reachable computation problem along with many other open source packages). It is observed that if we increase the number of template directions or sides of the initial set in the Support Function Algorithm(SFA) it increases the precision of reachability sets, which reduces the over-approximation error (leading to near actual computation). We also observed that as the directions in SFA increases the time taken to compute also increases. Thus, for complex hybrid systems with n-dimensional variables and with many directions, the task of reachability set computation is infeasible as it might take days or months or even years to compute for large time bound even with the best of the system tools available today. Moreover, in order to guarantee the safeness (to the extent of time bound) of any hybrid systems we must check or verify the system for large time bound.

We draw the conclusion that as we increase the number of directions we obtain precision in reachable states but the problem is the time taken for this increase directions with n-dimension and large time bound it is unacceptable as the reachability computation may take day, months or even years.

The major issues can be broadly categorise into:

1. Scaling the process of computing reachable sets for higher dimensional hybrid systems and for large time bounds.

2. Reachability computation suffers from *state space explosion problem* as described above. Therefore, to deal with the problem, appropriate abstraction model such as symbolic set representation for hybrid systems is required.

4 Problem to be taken up

In this research we intend to explore the Support Function technique as a basic convex *set representation*, which is used as a symbolic representation of reachable sets. It is considered to be one of the efficient technique to handle the state space explosion problem.

Our main focus in this research are :

1. To **speed up** the reachability computation process.

2. To **achieve precision**, some of the techniques may be :

   - Speeding up the algorithm which enable large number of directions.
   - Finding out efficient and appropriate directions for the given initial set.

5 Methodology/Expected outcome

A number of approaches would be explored and experimented elaborately to achieve speed-up and precision in reachability sets computation, some of the methods along with there expected outcomes are listed below:

5.1 Method 1: Implementing Parallel algorithms on multi-core CPUs:

5.1.1 Directions

Algorithm 1 performs, at each of its N iterations, a linear transformation on a vector *l*and the evaluation of the support functions ρ_I and ρ_V. The global complexity of Algorithm 1 is therefore $O(N(d^2 + C_I + C_V))$ where C_I and C_V denote the complexity of evaluating ρ_I and ρ_V , respectively. Let us remark that the complexity of Algorithm 1 is linear in the time horizon N and polynomial in d; which is comparable to the complexity of the most competitive algorithms [14]. Then, the computation of the tight over-approximations $\Omega_1, \Omega_2, ..., \Omega_N$ defined as intersections of the reachable sets Ω of r halfspaces has overall complexity of $O(rN(d^2 + C_I + C_V))$.

The advantage of Algorithm 1 is that it can be trivially parallelized into all the r directions simultaneously to compute the support functions to improve the computation time. The support function can be evaluated independently in the different directions $1, ..., r$. Thus, running the reachability analysis on$p$ processors makes the overall complexity drops to: $O(\left\lceil \frac{r}{p} \right\rceil N(d^2 + C_I + C_V))$ there by can achieve great speedup proportional to the number of processors.

Thus, with large number of processors available, greater speedup can be achieved easily. Moreover, with higher number of processors higher precision can easily be obtained by increasing the number of r directions there by reducing over-approximation error.

5.1.2 Iterations

We also observed that the complexity of the Algorithm 1, $O(rN(d^2 + C_I + C_V))$ can be reduced by parallelizing N iterations into p independent partitions, where p is the number of processors.

Thus, running the reachability analysis on$p$ processors makes the overall complexity reduces to $O(r \left\lceil \frac{N}{p} \right\rceil (d^2 + C_I + C_V))$.

Also with higher number of partitions the computation of reachable set is more **precise**, as for every partition the computation process begins with the exact initial input set, where as in the classical approach the process of computation of reachable set begins with just a single initial input set. In the present classical approach there exists the problem of wrapping effect, though the authors in their paper [14] have shown improvement by maintaining the factor τ as small as possible, which in turn increases the sampling time (time steps). Since the computation of the reachable set for any given location still begins with just a single initial input set so the process of consecutive convex hull of the two discrete polyhedra and then bloating them with radius of unit ball through out the continuous environment lead to wrapping effect.

Hence, with our approach of partitioning of the continuous environment into multiple partitions not only increases the time bound for reachable set but also lead us with multiple exact initial input set with multiple reduction in the wrapping effect, which will provide greater precision. With higher number of partitions, more multiple exact initial input set thus greater precision can be achieved. Also, as these partitions can be executed in parallel so reachability computation process will also speed up.

5.2 Method 2: An Efficient GPU implementation of the Simplex Method

We also observed that the complexity of the Algorithm 1 $O(rN(d^2 + C_I + C_V))$, is large with higher dimension of the hybrid systems, this is due to the large computation time involved in computing C_I and C_V (as observed by profiler Valgrind-callgrind/kcachegrind). Thus, parallelizing the computation of C_I and C_V which involves solving the Linear Programming(LP) problem (**Simplex Method**), great speedup can be achieved. We intend to use GPU programming approach to parallelize these computation and expect to see improvement.

Thus, running complexity reduces to $O(rN(d^2 + \left\lceil \frac{C_I}{p} \right\rceil + \left\lceil \frac{C_V}{p} \right\rceil))$. It is observed that (75-80)% of the time is involved in computing C_I and C_V, thus reducing computation time for the term $\left\lceil \frac{C_I}{p} \right\rceil$ and $\left\lceil \frac{C_V}{p} \right\rceil$ will greatly improve the whole computation process.

While exploring the GPU implementations of Simplex Method a number of optimising technique will be explored. Some of these could be, multiple LP solver over single GPU by appropriate partitions of SM cores over multiple LPs, optimising similar LPs over shared memory(also Cache memory if possible), e.t.c.

5.3 Method 3: Algorithm using a mixed approach of GPUs and CPUs Paralleliging

A number of intermixing of the GPUs and CPUs can be experimented one such naive approach is parallelizing the r number of directions using CPUs and accelerating the computation of C_I and C_V over GPUs. Thus, the complexity of the algorithm 1, $O(rN(d^2+C_I+C_V))$ reduces to $O(\left\lceil \frac{r}{p} \right\rceil N(d^2 + \lceil \frac{C_I}{P} \rceil + \lceil \frac{C_V}{P} \rceil))$, where p is the number of CPU's co-processors and P is the number of GPU cores.

Such a mixed approach will certainly improve the whole computation process by a great deal where by we can not only achieve speedup but can also gain in precision.

5.4 Method 4: Improvement on selection of Template-Directions

As mention in the paper [10] the Support Function of an input set S can be represented by over-approximation of a polyhedron in a template directions $D = \{\ell_1, \ell_2, ..., \ell_r\}$. The *template polyhedra*, is a polyhedra with faces whose normal vectors are given by D directions. A template polyhedron $P_D \subseteq R^n$ is a polyhedron for which there exist coefficients $b_1, ..., b_m \in R$ such that $P_D = \{x \in R^n | \bigwedge_{l_i \in D} l_i.x \leq b_i\}$. A template polyhedron can be represented by its coefficients b_i, which is useful when working with sets of template polyhedra.

The template directions considered in the tool SpaceEx [10] are box ($2n$) directions, octagonal ($2n^2$) directions and m uniform directions (which is as evenly as possible distributed over the unit sphere), where n is the dimensions of the hybrid systems.

We observed that selection of these template directions are not efficient as over-approximation error is large when the template direction is not normal to the initial polyhedra's (initial input set) faces as shown in the figure (2).

(a) Maximum Over-Approximation Error due to template box directions.

(b) Over-Approximation error eliminated due to appropriate selection of template directions.

Figure 2: Figure with green colour is the original initial input set and the region filled with red colour is the over-approximation error region due to selection of inappropriate template directions.

Thus, we proposed to select appropriate template directions so as to reduce the over-approximation error as small as possible to obtain precision in computing reachable set. In other words we shall select the template directions in such a way so as to represent a polyhedron that is as exact as that of the given initial input set S. This approach will not only result in precise computation of reachable set but will also helps us to speed-up the computation process by enabling us to choose only a limited number of template directions.

To select the appropriate template directions we begin by taking any given template directions (say box directions) and we take the directions one at a time and rotate each of this direction/vector by an angle δ until we find the best direction that is normal to the face of the initial input polyhedron. The notion best direction can be obtained by computing the support function over the initial input polyhedron with the rotated directions which returns the distances $D_1, D_2, ..., D_n$ among these distances the direction/vector producing the smallest distance is the best directions for the template directions.

The figure (2) above shows the rotation over two dimensional initial set for initial set with higher dimension the paper[**?**] gives a general n-Dimensional rotations of a point $\mathbf{x} = (x_1, x_2, ..., x_n)$ translated by a distance vector $\mathbf{d} = (d_1, d_2, ..., d_n)$ results to $\mathbf{x}' = (x_1', x_2', ..., x_n')$ which can be computed as $\mathbf{x}' = \mathbf{x}.T(\mathbf{d})$ and in an expanded matrix form in homogeneous coordinates it is denoted as

$$[x_1^{'}\ x_2^{'}\ ...\ x_n^{'}\ 1] = [x_1\ x_2\ ...\ x_n\ 1].\begin{bmatrix} 1 & 0 & ... & 0 & 0 \\ 0 & 1 & ... & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & ... & 1 & 0 \\ d_1 & d_2 & ... & d_n & 1 \end{bmatrix}$$

At this point of time we only assume that each of the component of the distance vector **d** can be computable independently for each direction/vector **x** and the best direction $\mathbf{x}^{'}$ can be obtainable form the template directions. The feasibility of the problem will be determined in due course of time.

6 Experiments

Using the C++ programming language we shall develop tool to implement the existing sequential algorithm to compute the reachable set of a hybrid systems provided with the initial input set and the dynamics of the hybrid systems. We will then implement our various approaches as described in the previous section and compare our results with the sequential algorithm to find out the improvement with regards to time as well as precision in reachability analysis. We will test the results on various Hybrid Systems models as available in the SpaceEx tool at http://spaceex.imag.fr such as Bouncing-Ball, Timed-Bouncing-Ball, Circle and Helicopter Controller[10]. We will also test with some of the benchmark model such Five Dimensional System[16] and Navigation[21], e.t.c.

We shall also be using a number of third party tools to speed up the process of development of hybrid systems for reachability set computation. Third party tools boost, GLPK(GNU Linear Programming Kit), e.t.c. We shall also compare the result with the variance of Linear Programming Kit such a Gurobi to see the performance difference.

Various standard profiling tools such as Valgrind's callgrind/kcachegrind and massif will also be used in order to profile the CPU time and memory usages during the reachability set computation.

7 Proposed Plan of work

The whole work have been divided into various list of goals and tasks as listed below:

- Goal 1: Development of hybrid systems tool
  - T1: Implement existing sequential algorithm using support function to compute reachable set for Continuous Environment of Hybrid Systems
  - T2: Using multi-core CPUs implement parallel (Directions) algorithm using support function to compute reachable set for Continuous Environment of Hybrid Systems
- Goal 2: Completing the development of hybrid systems tool
  - T3: Implement sequential algorithm using support function to compute reachable set for Discrete Transitions of Hybrid Systems
  - T4: Using multi-core CPUs implement parallel (Iterations) algorithm using support function to compute reachable set for Continuous Environment of Hybrid Systems
- Goal 3: GPU implementation of the Simplex Method
  - T5: Implementing GPU version of Simplex Method (Gimplex) and its variants.

  - T6: Implementing Gimplex method in the algorithm using support function to compute reachable set for Continuous Environment of Hybrid Systems.
  - T7: Parallel Algorithm using a mixed approach of GPUs and CPUs.

- Goal 4: Improvement on selection of Template-Directions.
  - T8: Implement sequential approach to compute *appropriate* template directions as described in the previous section.
  - T9: Implement parallel approach for task T8.
  - T10: Writing Thesis report.

The table 1 depicting the plan of work is as shown below.

Goal	Task	Sem–1	Sem–2	Sem–3	Sem–4	Sem–5	Sem–6
Goal 1	T1	X					
	T2	X					
Goal 2	T3		X				
	T4		X				
Goal 3	T5			X			
	T6			X	X		
	T7				X		
Goal 4	T8				X	X	
	T9					X	
	T10						X

Table 1: Showing Work plan describing semester-wise breakup of research tasks.

8 Significance of Research

The goal of the research is to improve the state-of-the-art towards computation of reachability analysis both in terms of speedup and precision there by making the task of reachability analysis feasible even for complex higher dimensional models of hybrid systems for large time bounds within a finite amount of time.

References

[1] R. Ray, *Reachability analysis of hybrid systems using support functions*. PhD thesis, 2009.

[2] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech," in *Hybrid Systems: Computation and Control*, pp. 258–273, Springer, 2005.

[3] W. Steiner, J. Rushby, M. Sorea, and H. Pfeifer, "Model checking a fault-tolerant startup algorithm: from design exploration to exhaustive fault simulation," *International Conference on Dependable Systems and Networks, 2004*, 2004.

[4] A. Chutinan and B. Krogh, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations," *Hybrid Systems: Computation and Control*, 1999.

[5] G. Frehse, R. Kateja, and C. Le Guernic, "Flowpipe approximation and clustering in space-time," *Proceedings of the 16th international conference on Hybrid systems: computation and control - HSCC '13*, p. 203, 2013.

[6] G. J. Holzmann, "The Model Checker SPIN," *IEEE Transactions on software engineering*, vol. 23, no. 5, pp. 279–295, 1997.

[7] K. G. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *International Journal on Software Tools for Technology Transfer*, vol. 1, pp. 134–152, 1997.

[8] S. Yovine, "Kronos : A Verifcation Tool for Real-Time Systems.," vol. 1, no. October 1997.

[9] T. A. Henzinger, P. H. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems," *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1997, pp. 110–122, 1997.

[10] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6806 LNCS, pp. 379–395, 2011.

[11] E. Lee, "Cyber Physical Systems: Design Challenges," in *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, 2008.

[12] R. Alur, "Formal verification of hybrid systems," in *Proceedings of Embedded Software (EMSOFT), 2011*, (Taipei, Taiwan), pp. 273–278, 2011.

[13] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler, "Recent progress in continuous and hybrid reachability analysis," in *IEEE International Conference on Control Applications*, 2006.

[14] C. Le Guernic and A. Girard, "Reachability analysis of linear systems using support functions," *Nonlinear Analysis: Hybrid Systems*, vol. 4, pp. 250–262, May 2010.

[15] A. Girard, C. Le Guernic, and O. Maler, "Efficient computation of reachable sets of linear time-invariant systems with inputs," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3927 LNCS, pp. 257–271, 2006.

[16] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Hybrid systems: computation and control*, pp. 291–305, 2005.

[17] Z. Han and B. Krogh, "Reachability analysis of large-scale affine systems using low-dimensional polytopes," *Hybrid Systems: Computation and Control*, pp. 287–301, 2006.

[18] I. Mitchell and C. Tomlin, "Level set methods for computation in hybrid systems," *Hybrid Systems: Computation and Control*, 2000.

[19] R. Alur, T. Dang, and F. Ivancic, "Predicate abstraction for reachability analysis of hybrid systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 5, no. 1, pp. 152–199, 2006.

[20] A. B Kurzhanski and P. Varaiya, "Ellipsoidal Techniques for Reachability Analysis," pp. 202–214, Springer, 2000.

[21] A. Fehnker, A. Fehnker, F. Ivancic, and F. Ivancic, *Benchmarks for Hybrid Systems Verification*, vol. 15213. Springer, 2004.