**[[Why does Astree scale up?]]**

Specific class of programs: that of reactive control-command software.

The programs for which ASTRÉE was designed are control-command software automatically generated from a graphical language à la SCADE. ASTRÉE aims at proving the absence of runtime errors in safety and mission critical synchronous control-command C code. Mission-critical synchronous control-command programs targeted by ASTRÉE feature very simple data-structures: mostly scalar numerical or boolean variables and statically allocated arrays or structures of constant size. No recursive nor dynamically allocated data-structures are used.

* Non citato, non dice nulla di interessante ai nostri fini.

**[[HySon: Set-based Simulation of Hybrid Systems]]**

Hybrid systems are a widely used model to represent and reason about control-command systems. [Hybrid systems are a widely used model to describe embedded systems which manipulate both continuous and discrete-time varying values] In an industrial context, these are often implemented in Simulink and their validity is checked by performing many numerical simulations in order to test their behavior with various possible inputs. The mixture of discrete and continuous-time in these systems provides them with a very useful expressive power, which enables one to model both discrete controllers and their continuous environment, but also makes them very difficult to study. Since they are often used to design critical systems, new verification tools are more and more needed. HySon is able to simulate a program given “imprecise” or “uncertain” inputs (for which we do not know the precise value but only an interval in which they lie), and computes a good approximation of the set of all possible Simulink executions, and this approximation takes in account errors due to the use of floating-point values to evaluate numerical formulas. To illustrate our method, we use a simple yet representative example, the uncertain bouncing pendulum. They use an improvement over interval arithmetic named affine arithmetic [9] which can track linear correlation between program variables. Experimental results include Ball, which is the bouncing ball with x0 = [10, 10.1], v0 = 15 simulated during 15s. Duration of the analysis in seconds = 0.068. --> forse l'esempio del pendolo potrebbe essere aggiunto? O qualcun altro di quelli citati nei benchmark?

**[[From control-command synchronous programs to hybrid automata]]**

Hybrid automata are often cited as the best model to describe and analyze control-command systems. In this article, we propose a framework that computes from a synchronous program an equivalent hybrid automata. Due to the development of the embedded software industry, there is a growing need for verification techniques for systems that mix discrete and continuous evolutions. Real time control-command programs are good examples of such hybrid systems.

A control-command software is a program that interacts with a continuous environment via sensors measuring some physical value and actuators modifying the dynamics of the physical values. Such programs often follow a loop "Read-Compute-Write", where "Read" means read values from the sensors, "Compute" means treat these values using numerical computations (often using floating point numbers) to chose the actuator to activate, and "Write" means actually changing the continuous dynamics using an actuator. Due to the real time constraint, this loop is executed at a specific frequency, which means that the "Read" and "Write" actions occur at specific, known time stamps. So, a real time control-command program implements a synchronous, sampled based controller that takes actions (i.e. changes actuators) at specific time instants and based on samples (i.e. inputs periodically read) of a continuous value.

In this article, we thus provide an automatic, semantics-preserving translation from H-Simple models to hybrid automata. This allows to easily integrate into an industrial development cycle the verification techniques developed for hybrid automata, and thus increase the confidence one can have in a control-command software. We believe that this work can then be used to verify, using model-based techniques, safety properties on control-command programs. The use of more abstract interpretation based techniques for the verification of the program are part of our future work.

**[[Proving the Correctness of the Implementation of a Control-Command Algorithm]]**

We leave the notion of "a program correctly controlling its environment" voluntarily vague as it may cover many properties. For example, one may want to prove the stability of the continuous trajectories under the influence of the program (which was mainly studied for switched dynamical systems [19, 29]), or one may try to prove that some region of the continuous state space is reached (which is the main goal of the existing analysis techniques on hybrid automata or equivalent models [18, 23]). We consider the so-called “two tanks system", well known from the hybrid system community. The main contribution of this paper is to provide a method for computing discrete trajectory invariants for control-command program given in the formalism presented in this section. While existing analysis techniques for hybrid systems are specialized for proving a certain kind of property on high-level models, this work focuses on inferring the most general invariants on code-level models. The state of the continuous environment is given by a vector of real values (for the continuous variables) and the continuous mode that governs its evolution. Continuous variables are abstracted by boxes. We explain how, given an abstraction of the program as a set of octrees, we compute an abstraction of the evolution of the continuous environment as a cyclic sequence. Benchmarks: Heater [0; 5] --> 0.071 Navigation [3:5; 3:6] x [3:5; 3:6] --> 5.456 Two-tanks [5; 5] x [6; 6] --> 32.9 Come nell'articolo di Hyson, mi sembra che gli intervalli dei valori iniziali siano ridottissimi!! Noi consideriamo uno spazio iniziale molto piú ampio. Inoltre diamo la risposta su quale sottoinsieme dello spazio iniziale dá risposte positive. (qui invece ti dicono solo una astrazione dello stato dopo tot iterazioni).

Our analysis relies on the domain of intervals: we use them as the abstraction of the physical environment and we use interval analysis techniques for computing the octrees that serve as a program abstraction. Of course, using relational domains can improve the precision of the analysis: we plan to investigate the guaranteed integration of ODEs using zonotopes.

**[[HyTech: A Model Checker for Hybrid Systems]]**

A hybrid system is a dynamical system whose behavior exhibits both discrete and continuous change. A hybrid automaton is a mathematical model for hybrid systems, which combines, in a single formalism, automaton transitions for capturing discrete change with differential equations for capturing continuous change.

Model checking can be used to provide more than a mere “yes" or “no" answer to the question of whether a system satisfies a correctness requirement. HyTech provides also diagnostic information that aids in design and debugging. If a system description contains design parameters, whose values are not specified, then HyTech computes necessary and sufficient constraints on the parameter values that guarantee correctness. A simple thermostat is used as a running example.

Applicato a molti benchmarks ma non c'é tabella con risultati e performances.

**[[The algorithmic analysis of hybrid systems]]**

We present a general framework for the formal specification and algorithmic analysis of hybrid systems. A hybrid system consists of a discrete program with an analog environment. We model hybrid systems as finite automata equipped with variables that evolve continuously with time according to dynamical laws. For verification purposes, we restrict ourselves to linear hybrid systems, where all variables follow piecewise-linear trajectories. We provide decidability and undecidability results for classes of linear hybrid systems, and we show that standard program-analysis techniques can be adapted to linear hybrid systems. In particular, we consider symbolic model-checking and minimization procedures that are based on the reachability analysis of an infinite state space. The procedures iteratively compute state sets that are definable as unions of convex polyhedra in multidimensional real space.

The purpose of this paper is to demonstrate that standard program-analysis techniques can be adapted to hybrid systems. Future work is necessary to improve both the cost and the scope of our approach. The cost can be improved by designing efficient algorithms for representing, comparing, manipulating, and approximating state sets. The scope can be improved by identifying other classes of hybrid systems to which semidecision procedures based on reachability analysis apply.

**[[Verification of linear hybrid systems by means of convex approximations]]**

We present a new application of the abstract interpretation by means of convex polyhedra, to a class of hybrid systems, i.e., systems involving both discrete and continuous variables. The result is an efficient automatic tool for approximate, but conservative, verification of reachability properties of these systems.

We consider a class of linear hybrid automata where guards are (general) linear conditions, actions are assignments of linear expressions, and derivatives are (symbolic) constants subject to linear inequalities. This class is general enough to handle many practical problems, and the linear restrictions allow easy symbolic analysis. Usa convex polhyedron (convex analysis method). (noi invece gestiamo non-convex)

**[[Discrete abstractions of hybrid systems]]**

Formal analysis of hybrid systems is concerned with verifying whether a hybrid system satisfies a desired specification, like avoiding an unsafe region of the state space. The formal analysis of large scale hybrid systems is tipically a very difficult process due to the complexity and scale of the system.

**[[Reachability analysis of hybrid systems via predicate abstraction]]**

This paper presents algorithms and tools for reachability analysis of hybrid systems by combining the notion of predicate abstraction with recent techniques for approximating the set of reachable states of linear systems using polyhedra. There has been increasing research on developing techniques for automated verification of hybrid (mixed discrete-continuous) models of embedded controllers. The state-of-the-art computational tools for model checking of HS are of two kinds, one based on rectangular inclusion, the other on polhyedra or ellipsoids. Scalability remains a challenge. For the sake of efficiency, attention restricted to linear expressions and linear continuous dynamics, possibly with bounded input. Our verifier performs on-the-fly search of the abstract system by symbolic manipulation of polhyedra. Convex hull. Foundations and initial prototype implementation for automated verification of safety properties of hybrid systems by combining ideas from predicate abstraction and polhyedral approximation of reachable sets of linear continuous systems. The number of abstract states grows exponentially. Rely on the intuition of the designer to obtain the initial set of predicates.

**[[Hybrid Systems: Modeling, Analysis and Control]]**

REVIEWER: Moreover, there also exist a huge amount of previous work on the related topic of hybrid systems (note that the bounding ball example from Sec. 3 actually appears as an example of hybrid system in course notes [c]). --> yes, but their example is simpler since they only consider vertical motion (therefore only two variables: vertical velocity and vertical position) while we consider also horizontal motion (doubling the number of variables).

Today most of the cost in control system development is spent on ad-hoc, prohibitively expensive systems integration, and validation techniques that rely almost exclusively on exhaustively testing more or less complete versions of complex nonlinear control systems.

[[BOXES]]

Developing correct and efficient disjunctive refinement is challenging. Boxes abstract domain – a refinement of the well-known Box (or Intervals) domain with finite disjunctions. An element of Boxes is a finite union of boxes, i.e., expressible as a proposi-tional formula over upper- and lower-bounds constraints. Our implementation is symbolic, and weds the strengths of Binary Decision Diagrams (BDDs) and Box. The complexity of the operations (meet, join, transfer functions, and widening) is polynomial in the size of the operands. Empirical evaluation indicates that the performance of Boxes is superior to other existing refinements of Box with comparable expressiveness.

Most common domains represent convex sets – i.e., conjunctions of linear constraints. There are several standard ways to build a disjunctive refinement. The simplest one is to allow a bounded number of disjuncts but it does not scale to a large number of disjuncts. The finite powerset construction [1, 2] represents disjunctions with finite sets and does not bound the number of disjuncts. If the base domain is finite, like in predicate abstraction [10], Binary Decision Diagrams (BDDs) [5] over the basis (i.e., predicates) of the base domain is thenatural choice for the completion. In this paper, we present a new abstract domain, Boxes, that is a disjunc-tive refinement of the well-known Box [7] (or Intervals) domain. That is, each value of Boxes is a propositional formula over interval constraints.

Each variable is bound to a unique dimension. Note, however, that the size of an LDD over V is in the worst case exponential in |V |.

The finite powerset of Box [1, 2], which we call PowerBox, is the main alternative to Boxes as a refinement of Box. An advantage of a finite powerset construction is its applicability to any base domain. However, this makes it hard (if not impossible) to leverage the power of domain-specific data structures. In contrast, our Boxes implementation is based on a specific data-structure – LDDs – but does not extend to other base domains.

LDD Boxes performs better than PPL Boxes, while maintaining a similar precision level. Boxes is a symbolic abstract domain that weds dis-junctive refinement of Box with BDDs. An area of future work is to study applicability and scalability of Boxes in a practical software verification setting. In particular, Boxes offers a promising platform for combining model-checking and abstract interpretation.

[[Sweeping line]]

We expose how concept of the sweeping line can be used to represent elements of a numerical abstract domain of boxes, which is a disjunctive refinement of a well known domain of intervals that allows finite number of disjunctions. We provide a detailed description of the representation along with standard domain operations algorithms. Furthermore we introduce very precise widening operator for the domain.

This paper covers the same area as the one by Gurfinkel and Chaki—we propose a different approach to the same domain of boxes. Our construction is more generic — the implementation of the domain that uses LDD’s can be regarded as an optimisation of the technique presented in this paper. We plan to check how the new widening operator behaves in practise and investigate what are the best values of the initial set of special points. [no benchmarks whatsoever]

* Ho deciso che non lo citiamo questo, é poco serio come paper.

[[Reachability Analysis of Hybrid Systems using Support Functions]]

REVIEWER: Although these works target real-valued functions, some ideas can surely be useful to analyze float programs, in particular partitioning techniques (for instance: [a]). --> *sinceramente non capisco in cosa ci potrebbe servire.*

* Non lo citerei.