Relazione Scientifica Annuale sull’attività svolta nell’ambito dell’assegno di ricerca

Nominativo dell’assegnista di ricerca: Matteo Camilli

Titolo dell’assegno di ricerca: Analisi di sistemi software complessi in ambito distribuito

Specificare se si tratta di assegno di ricerca di tipo A o di tipo B: Tipo A

Docente referente: Carlo Bellettini

Durata del contratto da 01/09/2015 a 31/08/2017

Periodo di riferimento della relazione da 01/09/2015 a 31/08/2016

Obiettivi della ricerca:

- Studio e definizione di approcci distribuiti per analisi di software complessi
- Studio e definizione di tecniche di analisi e di verifica di sistemi software adattivi
- Studio e definizione di tecniche di analisi di reti di Petri temporizzatore
- Studio e definizione di tecniche di analisi di sistemi software real-time

Risultati della ricerca:

The research activity contributes within different branches in the area of formal methods in software engineering.

The first main contribution lies in the introduction of distributed verification techniques that make use of the rapidly emerging paradigm of cloud computing, where challenges and opportunities allow for new research directions and applications. Our main goal and contribution has been to provide users with a model checking framework which can be easily deployed in the cloud. In fact, departing from the current literature on distributed CTL model checking, we considered an important, sometimes understated, aspect: we have enabled a “push-button” operating mode in the context of distributed formal verification to remove, or dramatically lower, the costs of deploying applications into end-to-end solutions.

Another contribution were the introduction of a modeling and verification technique for self-adaptive systems. In particular we focused on providing functional requirements assurance for real-time self-adaptive systems at design-time. The verification of timing requirements is supported through timed properties, able to check that both functional aspects and adaptation comply with specific temporal deadlines. In addition, the framework supports interesting (timed) robustness properties, to ensure self-healing capability that represents a very important issue when dealing with real-time or even time-critical systems. The proposed framework has been implemented as a Java software tool, called ZAFETY (http://camilli.di.unimi.it/zaafety). The framework has been successfully used to model and verify the self-healing behavior of some interesting real-world time-critical systems such as a gas burner system, and an unmanned aerial vehicle system.
Concerning the analysis of Petri nets based formalisms, we introduced a novel analysis technique for Time Basic (TB) Petri nets based on the efficient construction of a coverability tree. The proposed algorithm performs an exploration of the reachability tree of the model and uses acceleration along the tree's branches to reach the "limit" symbolic states. In addition, it can prune branches that are covered by symbolic states on other branches. The new technique is able to cope with unboundedness for all those models whose temporal functions do not express dependency on "infinite" past events. This technique is more powerful than the previous existing one. We are now able to solve significant reachability problems for TB nets with non-urgent behavior, in particular coverability, boundedness and place-boundedness. The technique has been successfully applied in the practice on some interesting case studies.

Some preliminary results contributes also in the area of runtime verification. In particular we are currently in the process of evaluating a novel technique (and related framework MAHARAJA, http://camilli.di.unimi.it/maharaja) for runtime verification of real-time JAVA programs with respect to their specifications given in terms of TB nets. MAHARAJA exploits aspect oriented programming to observe code execution and trigger the conformance verification at runtime. Our technique can be also used to perform testing activities by JUnit test cases, in order to discover both modeling and implementation faults. We believe that our approach represents a viable technique for checking real-time requirements of JAVA programs with respect to their formal specifications. The tool has been successfully used to monitor different real-time benchmarking case-studies and our experiments show that the monitoring overhead is bounded, although it could be further reduced by replacing AspectJ with more efficient bytecode transformation techniques. We shown that our monitoring framework uses negligible auxiliary memory. Moreover, the limited detection latency allows the system to be promptly recovered after a failure.

Attività svolte:

- Distributed formal verification

The recent extensive availability of "cloud" computing platforms is very appealing for the formal verification community. In fact, these platforms represent a great opportunity to run massively parallel jobs and analyze "big data" problems, although classical formal verification tools and techniques must undergo a deep technological transformation to take advantage of the available powerful architectures. We developed a distributed approach to verification of Computation Tree Logic (CTL) formulas on very large state spaces. The approach exploits and integrates our parametric state-space builder so called Mardigras (http://camilli.di.unimi.it/mardigras), we designed to ease the adoption of "big data" platforms.

The whole framework, composed by the state-space builder and the model-checker, employs Hadoop Map-Reduce as its computational engine and can be easily specialized to deal with verification of CTL formulas on huge state spaces generated from different formalisms (e.g., Petri Nets, Process Algebras etc.). We have already exploited it successfully to perform analyses of several Petri Nets benchmarking examples. The Map-Reduce programming model, which has become the de facto standard for large scale data-intensive applications, has provided researchers with a powerful tool for tackling big-data problems in different areas. Although new powerful cloud technologies are rapidly emerging, some Map-Reduce features seem to be tailored to backward/forward explicit state-space exploration. We firmly believe that explicit model checking could benefit from a MapReduce based approach.

Beside our model-checking engine, we studied the correctness and the complexity of the verification algorithms. Moreover we had a practical experience with some benchmarking Petri net models. The outcomes of several tests show the convenience of the proposed approach.
Specification and Verification of Self-adaptive Systems

Self-adaptive software systems are able to autonomously adapt their behavior at run-time to react to internal dynamics and to uncertain and changing environment conditions. Formal specification and verification of self-adaptive systems are tasks generally very difficult to carry out, especially when involving time constraints. In this case, in fact, the system correctness depends also on the time associated with events.

During our research activity in this area, we developed a formal framework to specify and verify the behavior of real-time self-adaptive systems. In particular, our main target systems are self-adaptive systems that exhibit a self-healing behavior. We define a specification formalism based on the Time-Basic Petri nets (or simply TB nets), a particular timed extension of Petri nets. The proposed formalism, called Zone-based TB Petri nets, provides some enhancements to the TB nets formalism to model self-adaptive systems with real-time constraints. In particular, we adopt and extend the classical adaptation models (i.e., one-point adaptation, overlap adaptation, and guided adaptation) to realize self-adaptation (SA) with temporal constraints in TB nets. To this purpose, the proposed specification formalism supports separation of concerns by allowing dividing the system's TB net model into zones. The proposed framework supports a verification technique for checking timed events through the symbolic execution of the system's TB net, thus allowing also the verification of timed adaptation. Zones of the TB net identified during the modeling phase can be used as modules (TB subnets) either in isolation, to check intra-zone properties, or all together, to check inter-zone properties of the overall system model. The framework has been implemented as a Java software tool, called ZAFETY (http://camili.di.unimi.it/zafety), and validated for modeling and verifying the self-healing behavior of some real-world time-critical systems.

Summarizing, in the context of formal modeling of self-adaptive systems with real-time constraints, we contribute with the following key aspects: (i) definition of a formalism for modeling self-adaptive behavior with real-time constraints; (ii) modular and incremental modeling and verification process due to separation in zones (e.g., adaptation/functional logics); (iii) verification technique for checking adaptation requirements satisfaction, including timed adaptation; (iv) approach able to map zones defined upon the model into regions of the system state space, thus allowing the identification of different behaviors (normal behavior, undesired behavior, adaptive behavior, etc.) upon the state space structure, and the verification of properties of interest with respect to SA that typically map to transitions between different zones.

Petri nets Analysis

The ability of analyzing unbounded Petri net models is a challenging issue. Naive state-space exploration techniques are doomed to fail on infinite sets of reachable states. A consolidated approach to tackle unboundedness is coverability analysis, that (under some assumptions) consists of building a finite over-representations of the model's state space. Some variations of the classical Karp & Miller (K&M) algorithm have been successfully proposed to extract (minimal) coverability sets from Place/Transitions (P/T) nets and other formalisms recognized as strictly-monotonic transition systems. Several important problems such as coverability (i.e., is a marking covered by any reachable one?), boundedness (i.e., is the set of reachable states finite?), place-boundedness (i.e., given a place p, is it possible to bound the number of tokens in p in any reachable state?), and semi-liveness can be decided on a coverability set.

When considering time PNs extensions representing real-time systems, (e.g., Timed PNs, or Time Basic PNs) coverability analysis is complicated by the presence of temporal constraints. In particular in the case of Time Basic Petri nets (TB nets), since tokens carry temporal information it is not generally possible to introduce \( \omega \) symbols (standing for any positive, possibly infinite, number of tokens), without loosing relevant information about the system's
behavior. A symbolic reachability graph for TB nets (TRG) has been recently introduced. This technique is characterized by the ability of locally recognizing some tokens as “time anonymous” (TA), meaning that the associated time information does not influence the model’s evolution.

Our research activity gave rise to a novel analysis technique for TB nets based on the efficient construction of a coverability tree, somehow inspired by the K&M algorithm. It copes with unboundedness caused by the accumulation of TA tokens in some places, at the price of an acceptable loss of information on model timing. Similarly to K&M an acceleration procedure is used (which essentially computes the limit of an infinite sequence of transition firings), tailored to deal with symbolic states and anonymous timestamps. The coverability analysis technique for TB nets computes the forward set of symbolic states reachable from the initial one. It deals with infiniteness in two different directions: that due to real-valued timings, handled by symbolic states, and the topological unboundedness, which refers to places holding an infinite number of TA tokens. Dealing only with TA tokens’ unboundedness represents an acceptable limitation in the practice. In fact it just restricts the class of analyzable models to those which do not include actions depending on “infinite” past events.

Since the boundedness problem is generally undecidable for urgent timed extensions of Petri nets, we assume a weak time semantics (non-urgent behavior) meaning that the model transitions may choose to do not fire and let the time pass, even if this might lead to transition disabling.

- **Runtime Verification**

  Runtime verification has been increasingly employed as a lightweight approach to guarantee the correctness of real-time software systems. Event-based runtime verification is the monitoring of running programs to verify the generated events against the requirements. A particularly challenging aspect is the monitoring of real-time properties. In fact, safety critical software has very often strict temporal requirements and monitoring at runtime introduces overheads on the System Under Test (SUT) that may affect the correctness of the verified properties.

  We introduced a particular event-based runtime verification technique and the related software implementation as a Java framework, so called MAHARAJA (http://camili.ili.unimi.it/maharaja). This framework enables the monitoring of Java programs, by evaluating the conformance of the concrete implementation with respect to its formal specification given in terms of Time Basic (TB) Petri nets, a powerful temporal extension of Petri nets (PNs) for modeling distributed systems with real-time constraints. Although, descriptive formalisms are very popular in runtime verification, the adoption of an operational specification offers some advantages with respect to declarative specifications. In fact, operational models are usually easier to write and understand. Moreover, they usually allow a step-wise model refinement to be used to trace the relation between the specification and the implementation.

  The MAHARAJA framework allows the SUT to be instrumented by using simple Java annotations on methods, in order to link the implementation to its formal model. Then, at runtime, the execution of the events of interest triggers the conformance verification of real-time requirements. Rather than use heavy offline computation to try to predict the rate of the generation of possible invalid events, thus the maximum detection latency, we focus on maintaining our analysis tool as lightweight as possible. MAHARAJA operates on and in conjunction with the SUT and it performs data collection and processing asynchronously with the SUT execution. The monitor and the SUT run concurrently on different isolated CPU cores by using a buffer-based mechanism for communication. Our approach tries to bound the cost of executing the SUT instrumentation by adding a limited number of instructions executed upon the generation of possible invalid events. The runtime verification procedure operates using just an occurring event and the 1-step reachability set of the current model's state, thus using
limited extra memory. The tool has been applied to a number of benchmarking case studies and we experimentally evaluated the runtime overhead, making it possible for a system designer to reason about the timing constraints of the SUT. Our experiments results show that MAHARAJA overcomes the performance of the current state-of-the-art runtime verification tools for Java programs, introducing limited detection latency and limited monitoring overhead.

Prodotti della ricerca conseguiti: (in termini di pubblicazioni, brevetti, ...)


Submitted for publication:


La presente relazione, non contiene dati sensibili e dati giudiziari di cui all'art. 4, comma 1, lettere d) ed e) del D.Lgs. 30.6.2003 n. 196. Si autorizza la pubblicazione della relazione annuale sul sito web del Dipartimento.

Firmato (In Stampatello)

**NOME ............... MATTEO .......... COGNOME... CAMILLI ...........**

Data 05/03/2016

Il Responsabile Scientifico

[Firma]

L' Assegnista di Ricerca

[Firma]