Model Checking a Path
(Preliminary Report)

N. Markey\textsuperscript{1,2} and P. Schnoebelen\textsuperscript{2}

\textsuperscript{1} Lab. d’Informatique Fondamentale d’Orléans
Univ. Orléans & CNRS FRE 2490
markey@lifo.univ-orleans.fr
\textsuperscript{2} Lab. Spécification & Vérification
ENS de Cachan & CNRS UMR 8643
markey|phs@lsv.ens-cachan.fr

Abstract. We consider the problem of checking whether a finite (or ultimately periodic) run satisfies a temporal logic formula. This problem is at the heart of “runtime verification” but it also appears in many other situations. By considering several extended temporal logics, we show that the problem of model checking a path can usually be solved efficiently, and profit from specialized algorithms. We further show it is possible to efficiently check paths given in compressed form.

1 Introduction

Model checking, introduced in the early 80’s, has now become a widely used approach to the verification of all kinds of systems \cite{CGP99,BBF+01}. The name “model checking” covers a variety of techniques dealing with various subproblems: how to model systems by some kind of Kripke structures?, how to express properties in temporal logics or some other formalisms?, how to use symbolic techniques for dealing with large state spaces?, and, most importantly, how to algorithmically check that a model satisfies a property?

These techniques rest upon a solid body of foundational knowledge regarding the expressive power of temporal logics and the computational complexity of their model checking problems \cite{Sch03}.

In this paper, we consider the problem of model checking a single path. This problem appears in several situations, most notably in runtime verification \cite{Dru00,Hav00,FS01}. There are situations where thousands of paths are checked one by one, e.g. the Monte-Carlo approach for assessing the probability that a random run satisfies some property \cite{YS02,LP02}. Less standard situations exist: \cite{RG01} advocates using temporal logic for describing patterns of intrusive behaviors recorded in log files. Such a log file, where a series of system events are recorded, is just a long path on which the temporal formula will be evaluated.

We do not restrict to finite paths and also consider checking ultimately periodic paths (given as finite “lasso-shaped” loop). Checking a path is much simpler...
than checking a Kripke structure, so much so that the problem may appear trivial: using standard dynamic programming methods “à la CTL model checking”, a path can obviously be checked in bilinear, i.e. $O(|\text{model}| \times |\text{formula}|)$, time.

This may explain why the problem, while ubiquitous, has not been isolated and studied from a theoretical viewpoint. For example, it is not known whether checking a simple temporal formula over a finite path can be done more efficiently than with the “bilinear time” method, e.g. with memory-efficient algorithms in SC, or with fast parallel algorithms in NC. Indeed this open problem was only identified recently [DS02].

With this paper, we aim to show that the problem is worthy of more fundamental investigations. Of course, the problem is a generic one, with many variants (which temporal logic? what kind of paths?) and here we only start scratching its surface.

More specifically, we present results (some of them folklore) showing that

Checking a Path is Easier: As we show in this paper, model checking a path is often much easier than checking a Kripke structure. We exhibit examples of richly expressive temporal logics that allow polynomial-time algorithms for checking a single path, while checking all paths of a Kripke structure is highly untractable. It is even possible to achieve polynomial-time when checking compressed paths (i.e. exponentially long paths that are given and stored in compressed form).

Checking a Path Relies on Specific Techniques: These efficient algorithms rely on specific aspects of the problem. Checking a path definitely comes with its own set of notions, technical tricks, and conceptual tools. For example, all our algorithms for checking ultimately periodic paths rely on a specific reduction technique to checking some kind of short finite prefix of the infinite path.

Outline of the Paper. We define the basic problem of model checking $LTL$ formulae over finite or ultimately periodic paths (Sect. 2). This problem is still not satisfactorily solved, but we argue that its intrinsic difficulty is already present in the case of finite paths (Sect. 3). We then show that model checking a path is much easier than model checking a Kripke structure by looking at various rich temporal logics: the monadic first-order logic of order, the extension of $LTL$ with Chop, or the extension of $LTL$ with forgettable past (Sect. 4). We provide polynomial-time algorithms for the last two instances. Finally we look at the problem of checking paths given in compressed form (Sect. 5).

Related Works. Model checking a path is a central problem in runtime verification. In this area, the problem is seen through some specific practical applications, sometimes with an emphasis on online algorithms, with the result that the fundamental complexity analysis has not received enough attention.

Dynamic programming algorithms for checking finite and ultimately periodic paths are also used in bounded model checking [BCC+03]. In this area, the relevant measures for efficiency are not the classical notions of running time and