Model Checking Propositional Deontic Temporal Logic via a $\mu$-Calculus Characterization

Araceli Acosta$^1$, Cecilia Kilmurray$^2$, Pablo F. Castro$^{2,3}$, and Nazareno M. Aguirre$^{2,3}$

$^1$ Facultad de Matemática, Astronomía y Física, Universidad Nacional de Córdoba, Córdoba, Argentina
aacosta@famaf.unc.edu.ar

$^2$ Departamento de Computación, FCEFQyN, Universidad Nacional de Río Cuarto, Río Cuarto, Argentina
{ckilmurray,pcastro,naguirre}@dc.exa.unrc.edu.ar

$^3$ Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

Abstract. In this paper, we present a characterization of a propositional deontic temporal logic into $\mu$-calculus. This logic has been proposed to specify and reason about fault tolerant systems, and even though is known to be decidable, no tool realizing its corresponding decision procedure has been developed. A main motivation for our work is enabling for the use of model checking, for analyzing specifications in this deontic temporal logic.

We present the technical details involved in the characterization, and prove that the model checking problem on the deontic temporal logic is correctly reduced to $\mu$-calculus model checking. We also show that counterexamples are preserved, which is crucial for our model checking purposes. Finally, we illustrate our approach via a case study, including the verification of some properties using a $\mu$-calculus model checker.

1 Introduction

With the increasing demand for highly dependable and constantly available systems, being able to reason about computer systems behavior in order to provide strong guarantees for software correctness, has gained considerable attention, especially for safety critical systems. In this context, a problem that deserves attention is that of capturing faults, understood as unexpected events that affect a system, as well as expressing and reasoning about the properties of systems in the presence of such faults.

Various researchers have been concerned with formally expressing fault tolerant behavior, and some formalisms and tools associated with this problem have been proposed\textsuperscript{1,2,3,7,9,12,16,15,14,13}. A particular trend in formal methods for fault tolerance, that concerns the work in this paper, is based on the observation that normal vs. abnormal behaviors can be treated as behaviors “obeying” and “violating” the rules of correct system conduct, respectively. From a logical point of view, this calls for a deontic approach, since deontic operators are especially well suited to express permission, obligation and prohibition, and thus
to describe fault tolerant systems and their properties [6]. This idea has been exploited by various researchers in different ways (see for instance [6, 17, 8, 5]). In this paper, we are concerned with the approach taken in [5], where a propositional deontic logic (PDL) is introduced, and then extended with temporal logic features to express temporal behavior with a distinction between normative (i.e., non faulty) and non normative (i.e., faulty) behaviors, with straightforward applications to fault tolerance.

In the context of formal approaches to software development, it is generally recognized that powerful (semi-)automated analysis techniques are essential for a method to be effectively used in practice. In particular, the possibility of algorithmically checking whether a PDL formula, or a formula in its temporal extension DTL, holds for a given system is of great relevance for the take up of these logics as part of a formal method for fault tolerance. Fortunately, both PDL and its temporal extension DTL are known to be decidable [5]: a decision procedure for the logic DTL, based on a tableaux calculus, is proposed in [4]. However, the proposed decision procedure had a theoretical motivation, namely, proving that the logic was decidable; in fact, this tableaux calculus proved useful for investigating decidability and the logic’s complexity, but was not devised as part of a tool for formal verification. Because of this fact, no practical considerations were taken in the definition of this decision procedure, and it has not been implemented in a tool for the analysis of fault tolerant specifications.

In this paper, we are concerned with the definition of a decision procedure for PDL and its extension DTL, with the purpose of being used for automated verification. Our approach consists of characterizing PDL/DTL in $\mu$-calculus, and then use a $\mu$-calculus model checker in order to verify whether a given system satisfies a fault tolerance property expressed in PDL/DTL. We thoroughly present our characterization of PDL/DTL in $\mu$-calculus, and show how a fault tolerant system, captured by a deontic structure, can be analyzed for the satisfaction of PDL/DTL formulas, describing fault tolerant properties of the system. Moreover, we show that our translation from PDL/DTL into $\mu$-calculus is correct, in the sense that the model checking problem in PDL/DTL is soundly reduced to model checking in $\mu$-calculus. Moreover, we also show that counterexamples are maintained, meaning that every $\mu$-calculus counterexample, resulting from the verification of a translated property on a translated model, can be mechanically traced back to a counterexample of the original deontic temporal specification. Finally, we provide some experimental results using the Mucke $\mu$-calculus model checker [2], on a small case study illustrating how deontic structures capture systems with faults, and also illustrating our approach, as well as the details of our translation.

The paper proceeds as follows. In section 2 we present some preliminaries, including the syntax and semantics of PDL, as well as those of the $\mu$-calculus. Section 3 introduces our translation from the core logic PDL to $\mu$-calculus, and a proof of the correctness of the translation. Section 4 introduces DTL, consisting of PDL extended with CTL temporal operators, and Section 5 deals with the translation from DTL to $\mu$-calculus, including a proof of the correctness of this