**CTL* and ECTL* as fragments of the modal μ-calculus**

Mads Dam  
Department of Computer Science  
University of Edinburgh

1 Introduction

Due to its inherent combinatorial difficulties, concurrency is an area of computer science where formal and automated verification methods have proved themselves particularly valuable, for instance in detecting errors difficult or impossible to find by informal reasoning alone (c.f. Gries [16], Clarke and Mishra [7], Browne et al [3]). For programming errors to be exposed by exhibiting mismatch to formal properties, those properties must correctly reflect the intent of the programmer. The correctness of this representation may be obvious for a few very simple theories, but unfortunately the need for greater expressive power often seems to call for a corresponding sacrifice in transparency.

A case in point is the modal μ-calculus $L_\mu$ (Kozen [18]). This logic is obtained as an enrichment of a simple modal base logic, Hennessy-Milner logic (Hennessy, Milner [17]), by least and greatest fixpoints of formally monotone operators. The result is a very general branching-time temporal logic capable of expressing a wealth of properties related to for instance partial and total correctness, liveness, safety and fairness, that are of crucial importance in practical program verification (c.f. Bradfield, Stirling [2], Walker [30]). Indeed $L_\mu$ encompasses a great many well-known program logics such as PDL (Fischer, Ladner [14]), PDL-Δ (Streett [25]), linear-time temporal logic PTL (Gabbay et al [15]), CTL (Clarke, Emerson [5]), and CTL* (Emerson, Halpern [10]). This can be shown by providing constructive translations (c.f. Kozen [18], Emerson, Lei [12], Wolper [31]).

In fact the containment in all these cases is strict. A typical example of a property expressible in $L_\mu$ but not, for instance, in CTL* is a cyclic property such as "along any path, at all even moments, φ holds" (c.f. Wolper [32]). Properties such as these are necessary in general for modular reasoning (Lichtenstein et al [19]). Despite this additional expressive power, $L_\mu$ is decidable in deterministic exponential time, and thus essentially is not harder than PDL (Emerson,

---

1This work was supported by SERC Grant GR/F 32219.
Jutla [11], Fischer, Ladner [14]). Moreover, for closed formulas $L_\mu$ preserves the characterisation of bisimulation equivalence (Hennessy, Milner [17]) and thus provides a natural temporal logic for process calculi such as CCS (Stirling [23]). A model checker for checking $L_\mu$-properties against finite-state (CCS) processes due to Stirling and Walker [24] has been implemented in the Edinburgh Concurrency Workbench (Cleaveland et al [8]) and used in several case-studies such as mutual exclusion algorithms (Walker [30]) and communication protocols (Bruns, Anderson [4]).

Impeding the widespread practical use of $L_\mu$, however, is its lack of transparency: Already at the second level of alternation formulas can become highly unintelligible. And alternation is indeed needed to express for instance fairness properties. Consequently it is important to develop tools to aid users in manipulating, generating, and understanding $L_\mu$-formulas. Constructive translations such as those referred to above can be very useful for these purposes. They can be machine implemented to provide syntactical sugaring of $L_\mu$ properties. Moreover they can help also in understanding $L_\mu$ itself, provided they, and the results they produce, are sufficiently simple and intuitive. Most of the translations succeed very well in this: A first step towards a working understanding of $L_\mu$ is certainly to understand why the CTL-formula $\text{EF}X$ is translated into $\mu Y.X \lor \Box Y$.

In this respect CTL* (Emerson, Halpern [10]) is of particular interest. Beyond CTL, CTL* is capable of expressing properties such as $\text{EGF}\phi$ ("along some path $\phi$ holds infinitely often"), useful for dealing with fairness (Emerson, Halpern [10]), and in general arbitrary nestings and boolean combinations of linear and branching time connectives for which the task of finding equivalent $L_\mu$-formulations may present considerable difficulties. The previously only known translation of CTL* into $L_\mu$ is, however, rather indirect and not very transparent. It is obtained by composing Wolper’s unpublished translation of CTL* into PDL-$\Delta$ [31] with the translation of PDL-$\Delta$ into $L_\mu$ (c.f. Emerson, Lei [12]). It involves 5 stages: The first and second stages builds a tableau and derives from it a deterministic Muller automaton (c.f. Emerson and Sistla [13]). Third stage derives from this automaton an equivalent $\omega$-regular expression (c.f. McNaughton [20]). As the fourth stage a PDL-$\Delta$ formula is obtained, and finally this formula is translated into $L_\mu$.

In this paper we present a relatively simple and much more direct algorithm for translating CTL* into $L_\mu$. The idea is to represent a tableau directly as an equivalent $L_\mu$ formula. The notion of tableau used is fairly standard and closely related to those of e.g. Ben-Ari et al [1] and Wolper [32]. Their role is to decompose formulas according to their structure, and to detect recursion in the natural way of terminating when a node is repeated. The problem is to use the connectives of $L_\mu$ as an external means of characterising tableaux, and in particular to use least and greatest fixpoints to classify loops. The solution involves an analysis of the admissible ways of "regenerating" nodes, using the terminology of Streett and Emerson [26].

An alternative way of allowing cyclic properties to be expressed, is to include