COLLEGE OF COMPUTING AND INFORMATION SCIENCES
DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING AND INFORMATICS TECHNOLOGY

April 21, 2017

NAME: OKIRING PAUL -214016765 - 14/U/13973/EVE
NAME: MWESIGYE MICHAEL -214019414 - 14/U/10401/EVE
NAME: OKELLO JOHN PAU -215005517 - 15/U/11991/EVE
NAME: WANGIRA JOHN - 214017326 -14/U/15832/EVE

CONCEPT PAPER-GROUP202

BRANCHING TEMPORAL LOGICS,AUTOMATA AND GAMES

1 INTRODUCTION

Model checking has emerged as a powerful method for the formal verification of programs. Temporal logics such as CTL (computational tree logic) and CTL* are widely used to specify programs because they are expressive and easy to understand. Given an abstract model of a program, a model checker (which typically implements the acceptance problem for a class of automata) verifies whether the model meets a given specification. A conceptually attractive method for solving the model checking problem is by reducing it to the solution of (a suitable subclass of) parity games. These are a type of two player infinite game played on a finite graph.

1.1 Is automatic program verification important?

P-ogram verification is a critical problem in computer science. Many believe that man- ual, informal verification is acceptably reliable and more practical than automatic formal verification. However, verification of distributed programs is difficult to achieve manually. Consider the following simple distributed algorithm. P1 and P2 are two processes that are executed on a processor. They share a common variable t, and each has a local variable yi,I = 1,2 that can be read by the other process. We assume that at each step exactly one process is active, i.e., it executes the current statement. P1 and P 2 sharp a common resource (printer or disk) and the algorithm is meant to guarantee that at most one process uses the resource at a given time. We say that the process is in the critical section when it uses the resource. L 3 and M 3 are the critical sections for P1 and P2 , respectively. The reader is challenged to verify whether the following algorithm satisfies two properties: 1. "Mutual exclusion" is guaranteed: it is never true that P is at L3 and P2 is at M 3. 2. Liveness is guaranteed: always if P (P 2) is at L, (MI) (requesting the resource), then eventually P1 (P 2) will be at L3 (M 3) (getting the resource).

2 PROBLEM STATEMENT

Given a model of a system, exhaustively and automatically check whether this model meets a given specification. Typically, one has hardware or software systems in mind, whereas the specification contains safety requirements such as the absence of deadlocks and similar critical states that can cause the system to crash. Model checking is a technique for automatically verifying correctness properties of finite-state systems. In order to solve such a problem algorithmically, both the model of the system and the specification are formulated in some precise mathematical language. To this end, the problem is formulated as a task in logic, namely to check whether a given structure satisfies a given logical formula. This general concept applies to many kinds of logics and suitable structures. A simple model checking problem is verifying whether a given formula in the propositional logic is satisfied by a given structure.

3 BACKGROUND

The satisfiability problem for branching-time temporal logics like CTL*, CTL and CTL+ has important applications in program specification and verification. Their computational complexities are known: CTL* and CTL+ are complete for doubly exponential time, CTL is complete for single exponential time. Some decision procedures for these logics are known; they use tree automata, tableaux or axiom systems. Automata-theoretic approaches. As much as the introduction of CTL* has led to an easy unification of CTL and LTL, it has also proved to be quite a difficulty in obtaining decision procedures for this logic. The first procedure by Emerson and Sistla was automata-theoretic [ES84] and roughly works as follows. A formula is translated into a doubly-exponentially large tree automaton whose states are Hintikka-like sets of sets of sub formulas of the input formula. This tree automaton recognizes a superset of the set of tree models of the input formula. It is lacking a mechanism that ensures that certain temporal operators are really interpreted as least fix points of certain monotone functions rather than arbitrary fix points. Other approaches. Apart from these automata-theoretic approaches, a few deferent ones have been presented as well. For instance, there is Reynolds' proof system for validity [Rey01]. Its completeness proof is rather intricate and relies on the presence of a rule which violates the sub formula property. In essence, this rule quantity over an arbitrary set of atomic propositions. Thus, while it is possible to check a given tree for whether or not it is a proof for a given formula, it is not clear how this system could be used in order to find proofs for given formulas.

4 MAIN OBJECTIVES

To show the connexions between the temporal logics CTL and / or CTL*, automata, and games.

4.1 OBJECTIVES

1. Representing CTL / CTL* as classes of alternating tree automata

2. Inter-translation between CTL / CTL* and classes of alternating tree automata

3. Using Buchi games and other subclasses of parity games to analyze the CTL / CTL* model checking problem

4. Efficient implementation of model checking algorithms

5. Application of the model checker to higher-order model checking

5 Scope

Our overall research is basically limited to the areas of Branching Temporary Logics, Automata and Games. The term Temporal Logic has been broadly used to cover all approaches to representation and reasoning about time and temporal information within a logical framework, and also more narrowly to refer specifically to the modal-logic type of approach introduced around 1960 by Arthur Prior under the name of Tense Logic and subsequently developed further by many logicians and computer scientists. Applications of Temporal Logic include its use as a formalism for clarifying philosophical issues about time, as a framework within which to define the semantics of temporal expressions in natural language, as a language for encoding temporal knowledge in artificial intelligence, and as a tool for specification, formal analysis, and verification of the executions of computer programs and systems.

6 Preliminaries

6.1 The temporal logic CTL

We consider the standard branching-time temporal logic CTL. For technical reasons, we use the operator U as a dual of the until operator for which the stop condition is not required to occur; and we suppose w.l.o.g. that formulas are given in positive normal form, i.e., negations are applied only to atomic propositions. Indeed, each CTL formula can be written in positive normal form by pushing the negations inside. Denition 1. Let AP = a,b,c,... be a nite set of atomic propositions. The set of CTL formulas is given by (where a AP): ::= a — a — — — AX — EX — A[U] — E[U] — A[U] — E[U].

6.2 Alternating Buchi PushDown Systems

An Alternating Buchi PushDown System (ABPDS for short) is a tuple BP = (P,,, F), where P is a nite set of control locations, is the stack alphabet, F P is a nite set of accepting control locations and is a function that assigns to each element of P   a positive boolean formula over P .

6.3 PushDown Systems

A PushDown System (PDS for short) is a tuple P = (P,,,]), where P is a nite set of control locations, is the stack alphabet,   (P) (P) is a nite set of transition rules and]   is a bottom stack symbol.

6.4 Recursion Schemes

In higher-order model checking, recursion schemes are typi- cally used as simply-typed grammars for constructing possibly innite term-trees or ranked trees. Types are dened by the grammar A ::= o — A B. By convention, arrows associate to the right; thus every type can be written uniquely as A1 An o, for some n 0. We dene the order of a type: ord(o) := o and ord(A B) := max(ord(A) + 1,ord(B)). Intuitively the order of a type measures how deeply nested it is on the left of the arrow.

7 Significances

Advantages of the game-based approach. The game-theoretic framework uniformly treats the standard branching-time logics from the relatively simple CTL to the relatively complex CTL. It yields complexity-theoretic optimal results, i.e. satisfiability checking using this framework is possible in exponential time for CTL and doubly exponential time for CTL+. Like the automata-theoretic approaches, it separates the characterization of satisfiability through a syntactic object (a parity game) from the test for satisfiability (the problem of solving the game). Thus, advances in the area of parity game solving carry over to satisfiability checking. Like the tableaux-based approach, it keeps a very close relationship between the input formula and the structure of the parity game thus enabling feedback from a (counter-)model or applications in specification and verification. Satisfiability checking procedures based on this framework are implemented in the MLSolver platform [FL10] which uses the high-performance parity game solver PG-Solver [FL09] as its algorithmic backbone.

8 Methodologies

The decision procedures for these logics are known;

The tree automatas a tree automaton is a type of state machine that deals with tree structures,rather than the strings of more conventional state machines.

Tableaux this is a decision method for propositional temporal logic that can be used to formalize reasoning about time and temporal relations.

Axiom systems An axiomatic system is any set of axioms from which some or all axioms can be used in conjuction to logically derive theorems. Due to the ability of temporal logic to allow us to make deductive arguments about not only what is,but what was , what will be and what will always be, this has led to the application of the logics to specifications and verification of reactive and concurrent programs and systems i.e games. The key temporal patterns of importance in specifying such programs are:

Liviness properties or eventualities which ensure a specific precondition is intially satisfied. Then a desiarable state is eventually reached.

Safety or Invariance properties this ensures that a specific precondition is intially satisfied, then undesirable state will never occur.

Fairness properties -Fairness requires that in a system where several processes sharing resources are run concurrently , they must be treated fairly by the program.

Artificial Intelligence Temporal reasoning has also been naturally combined with other well developed framework for AI such as the situation calculas(Pinto and Rieter 1995) and action theory (Lamport 1994)

Traditionally methods such as temporal arguments in which the temporaldimensions is captured by augmenting each time variable proposition or predicate with an extra argument place to be filled by an expression designating a time.

9 References

1.Feidmann, O. (2013). Satisfiability Games For Branching-Time Logics. University of .

2.Grumberg, O. (2008). 25 Years Of MOdel Checking. Grumberg, Orna, Veith, Helmut (Eds.).

3.Monkowski. (2015, May 20). Temporal Logic. Retrieved April 18, 2017, from plato.stanford.edu

4.Orna Kupferman, Moshe Y. Vardi, Pierre Wolper: An automata-theoretic approach to branchingtime model checking. J. ACM 47(2): 312-360 (2000). http://dx.doi.org/10.1145/333979.333987

5.Pinto,D. Rieter,T. (1995) Knowledge Representation, Reasoning, and the Design of Intelligent Agents. Cambridge University Press