The Relationship Between Temporal Logic Normal Forms and Alternating Automata

Clare Dixon[†], Alexander Bolotov[‡] and Michael Fisher[†]

[†] Department of Computer Science, University of Liverpool, Liverpool L69 7ZF,
{M.Fisher,C.Dixon}@csc.liv.ac.uk

[†] Harrow School of Computer Science, University of Westminster, Harrow HA1 3TP
A.Bolotov@westminster.ac.uk

Introduction

The connection between temporal logic and automata of different kinds is well known. A model for a propositional linear-time temporal logic formula, φ, is essentially a sequence of states where the propositions from φ are set to true or false such that the sequence of states and setting of propositions satisfies φ. This sequence can be viewed as an infinite word over subsets of the propositions in φ. Thus for any propositional linear-time temporal logic formula we can construct a finite automaton such that the automaton accepts exactly the sequence of states (infinite word) which satisfies the formula [7]. If the propositional linear-time temporal logic formula is unsatisfiable then the automata constructed is empty.

SNF (Separated Normal Form) is a normal form for representing propositional linear-time temporal logic (PLTL) formulae [6]. The normal form comprises formulae that are implications with present-time formulae on the left-hand side and (present or) future-time formulae on the right-hand side. The transformation into the normal form reduces most of the temporal operators to a core set and rewrites formulae to be in a particular form. The transformation into SNF depends on three main operations: the renaming of complex subformulae; the removal of temporal operators; and classical style rewrite operations. SNF has been used as the basis for a temporal resolution method [6] and for execution of temporal formulae [1]. In previous work we have considered the relationship between Büchi Automata and SNF [2].

An alternating automata is an automata that has both the power of existential and universal choice, similar in structure to an and/or graph. Alternating automata were considered in [3, 4] and have been studied in relation to temporal logic in [7] for example.

Alternating Automata

We define alternating automata following [7]. A is an alternating automata $A = (\Sigma, S, s_0, \rho, F)$ such that

- Σ is a finite non-empty alphabet;
- S is a set of states;
- $s_0 \in S$ is an initial state;
- $\rho : S \times \mathcal{P}(\Sigma) \to \mathcal{B}^+(S)$ is a transition function

- $F \subseteq S$ is a set of accepting states;

where $\mathcal{B}^+(S)$ are the set of positive Boolean formulae over S, i.e. Boolean formulae built from S using $\wedge$ and $\vee$. Runs in alternating automata are trees. In the following, given a tree τ, ϵ is the root node of τ, x is a node in τ and $|x|$ is the distance of x from the root, where $|\epsilon| = 0$. An W-labelled tree, for some set W is a tree, τ, and mapping t such that t maps nodes of τ to W. A *run* of A on an infinite word $w = a_0, a_1, \ldots$ is a possibly infinite S-labelled tree r such that $r(\epsilon) = s_0$ and the following holds. If $|x| = i$, $r(x) = s$ and $\rho(s, a_i) = \theta$, then x has children $x_1, \ldots x_k$ for some $k \leq |S|$ and $\{r(x_1), \ldots, r(x_k)\}$ satisfies θ. The run is *accepting* if every infinite branch in r includes infinitely many labels in F. Note the run can also have finite branches; if $|x| = i$, $r(x) = s$ and $\rho(s, a_i) = \textbf{true}$ then x does not need to have any children. However we cannot have $\rho(s, a_i) = \textbf{false}$ as **false** is not satisfiable. Thus every branch in an accepting run has to hit **true** or hit accepting states infinitely often.

The Translation Between SNF and Alternating Automata

We provide transformations from alternating automata into SNF and from SNF into alternating automata in [5]. We show that a set, R, of SNF clauses is satisfiable if and only if the alternating automata, A_R, constructed from R has an accepting run. Similarly an alternating automata A has an accepting run if and only if the set of clauses R_A constructed from A is satisfiable.

This is part of ongoing work looking at the relationship between SNF and other formalisms. We were prompted to investigate the connection between SNF and alternating automata due to and/or structure of SNF i.e. SNF is a conjunction of clauses which are essentially disjunctions. In particular, this result is useful as there is no direct method for checking the non-emptiness of an alternating automata. However non-emptiness can be checked by, for example, giving a translation into a (nondeterministic) Büchi automata (see for example [7]) and doing the non-emptiness check there. Here, we can use the translation provided in [5] and then apply temporal resolution to the clauses obtained. If the clauses obtained are unsatisfiable then this means there is no accepting run.

References

[1] H. Barringer, M. Fisher, D. Gabbay, R. Owens, and M. Reynolds, editors. *The Imperative Future: Principles of Executable Temporal Logic.* Research Studies Press, May 1996.

[2] A. Bolotov, M. Fisher, and C. Dixon. On the Relationship between w-Automata and Temporal Logic Normal Forms. *Journal of Logic and Computation*, 2002. To appear.

[3] J. Brzozowski and E. Leiss. Finite automata, and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.

[4] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *ACM Journal*, 28(1):114–133, 1981.

[5] C. Dixon, A. Bolotov, and M. Fisher. Alternating Automata and Temporal Logic Normal Forms. In preparation.

[6] M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, January 2001.

[7] M. Y. Vardi. An Automata-Theoretic Approach to Linear Temporal Logic. In *Proceedings of Banff'94*, Lecture Notes in Computer Science, pages 238–226. Springer-Verlag, 1994.