asCTL Model Checker

Matrics: 150003282 and XXX

November 2018

Abstract

1 Introduction

1.1 Structure

2 Logic and Model Specifications

2.1 Model

2.2 asCTL Formula

2.2.1 Action Sets

Where action sets were provided, an empty list was interpreted as the set of no transitions. If no list was given, any transition was allowed, as there were no restraints on the operation.

Action sets were treated as constraints on which paths to consider, rather than further logical conditions to check. For example, $\forall G_A a$ would return true for a state from which all onwards paths of transitions from A gave states where a was true, and would not be violated if there was a transition from the state that was not included in A. Only the satisfiability of the state formula were verified.

To verify that no transitions other than those in A were possible, the condition $\exists_A F_B \texttt{True}$ would be verified, where B is the set of all actions not in A (or the set of all undesirable actions).

2.2.2 Strong Until

$$a_A \bigcup_B b$$

a holds until b. If a transition from B is possible and leads to a state where b holds, the expression is true. If not, transitions from A to states where a is true are made until it is possible. If such a path exists, the expression holds. If a final transition cannot occur, and no transitions from A are possible, the expression is false. A path of A transitions resulting to a cycle also fails to satisfy the expression.

This implements strong until, as the only accepted paths are those that end in a transition from B to a state where b holds, and all other transitions are from A and end in states where a holds.

If B is not specified, the expression can be true before the first transition, if the initial state satisfies b. If B is given, at least one transition must occur.

2.2.3 Eventually

$${}_A F_B a \equiv \texttt{True}_A \bigcup_B a$$

Eventually is true if a transition from B ends in a state where a is true, and all other transitions are from B. This is logically equivalent to the given strong until clause.

As with strong until, eventually can be true before the first transition if and only if B is not specified.

2.2.4 Weak Until

$$a_A W_B b$$

Weak until is true for paths where a transition from B leads to a state where b is satisfied, and all transitions until then are from A and end in states satisfying a. It also accepts cyclic paths of transitions from A in which every state satisfied a, and any path leading to a state from which no transition in B gives a state where b is true and no transitions from A are possible.

As with strong until, if B is not specified the expression can be true before the first transition, but if it is, at least one transition is needed.

2.2.5 Always

$$G_A a \equiv a_A W_{[]} \texttt{False}$$

Always holds if all transitions from A lead to states where a is true, and all onwards transitions from A continue to lead to states where a is satisfied. It fails only for paths of transitions from A containing a state the does not satisfy a.

It is equivalent to the given weak until clause.

2.2.6 Next

$$X_A a$$

Next is true for a path if there is a transition in A from the current state to a state where a is true.

2.2.7 Existential and Universal Qualifiers

2.3 Constraints

Constrains narrow down the search space by forbidding certain transitions.

In each node of the search tree, the constraint must hold. A child constraint can then be derived that applies to branches from that node.

A child constraint is derived by simplifying the stable values of the constraint. For example, an atomic proposition applies only to the current state, so in child constraints will remain stable as either true or false, depending on whether or not it holds currently.

Constraint -> child Bool. Const. -> same constant Atomic Proposition -> bool (is it true in the current state) Neg/And/Or -> same operator on child constrains of sub expressions Exists -> evaluated to a boolean For All -> evaluated to a boolean or another For All

3 Implementation

4 Testing

5 Results

References