CS4052 Coursework 2: A Simple Model Checker

Juliana Bowles

October 26, 2016

This practical is worth 50% of the coursework grade for this module and is **due 21:00 on Wednesday 23rd November, 2016.** The practical is to be done in **groups of two**. Groups have been set up and a list is available on studres.

The practical involves developing a simple model checker for an *action* and *state-based logic* $asCTL$ interpreted over transition systems. The model checker supports fairness in the sense that you can add any constraints to the model to restrict the set of paths being considered for verification. To make it easier, some components (such as a parser and a model builder) are provided. Possible **Extensions** are included at the end but feel free to consider additional ones. You have to complete an extension activity to achieve a **mark above 17**.

Logic $asCTL$

In the lectures we have seen two state-based logics, namely LTL and CTL. Neither of the logics is suitable for making statements about the sequences of actions carried out before reaching a particular state. One example of a property we may want to be able to express is *it is always the case that after action* α *or* β *occurs the system is in a state where* p *holds.* Or *the system will eventually reach a goal state by only performing legal actions.*

Similarly to CTL, in $asCTL$ we distinguish between state and path formulae. Let Φ be a state formula, ϕ a path formula, $p \in AP$ an atomic proposition, $\alpha \in Act$ an action, and $A, B \subseteq Act$ subsets of actions.

The *action* and *state-based logic* $asCTL$ has a grammar defined as follows.

State formulae are given by:

$$\Phi ::= true \mid p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \exists\phi \mid \forall\phi$$

A *path formula* is given by:

$$\phi ::= \Phi \ \mathcal{U}_B \ \Phi \mid \Phi_A \mathcal{U}_B \ \Phi \mid \Phi_A \mathcal{U} \ \Phi \mid \Phi \ \mathcal{U} \ \Phi$$

The temporal operator *until* in a path formula is indexed by $A, B \subseteq Act$. Let a path be given by

$$\pi \equiv s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \ldots s_{i-1} \xrightarrow{\alpha_i} s_i \ldots$$

The intended meaning of a path formula over a path π is as follows:

$\pi \models \Phi_1\,{}_A\mathcal{U}_B\ \Phi_2$	holds iff there is a state s_i in π such that $s_i \models \Phi_2$, $\alpha_i \in B$, and until then all states s_k with $0 \leq k < i$ $s_k \models \Phi_1$ and $\alpha_1, \ldots, \alpha_{i-1} \in A$.

In other words, just before Φ_2 holds a B action must occur, and until then Φ_1 holds and only A actions can occur. Notice that A, B are subsets of the overall sets of actions. You need to think about what it means (or in your opinion it should mean) for A or B to be $\emptyset$ or Act.

If one of the sets is missing, like in $\Phi_1\ \mathcal{U}_B\ \Phi_2$ or $\Phi_1\,{}_A\mathcal{U}\,\Phi_2$ then we are effectively imposing fewer restrictions:

$\pi \models \Phi_1\ \ \mathcal{U}_B\ \Phi_2$	holds iff there is a state s_i in π such that $s_i \models \Phi_2$, $\alpha_i \in B$, and until then all states s_k with $0 \leq k < i$ $s_k \models \Phi_1$.
$\pi \models \Phi_1\,{}_A\mathcal{U}\ \ \Phi_2$	holds iff there is a state s_i in π such that $s_i \models \Phi_2$, and until then all states s_k with $0 \leq k < i, s_k \models \Phi_1$.

Finally, $\Phi_1\ \mathcal{U}\ \Phi_2$ has the same meaning as in CTL.

You will need to think about and define equivalences between formulae and the meaning of all uses of the until operator. Further, how can you for instance obtain path formulae with X (next), F (eventually) and G (always), and what kind of action-indexed operators X, F and G make sense? In the case of the next operator, for instance, we may only allow $X_B\Phi$ to denote that in the next state Φ holds and a B action occurred.

Main Task

Develop a model checker for the logic $asCTL$, an action and state-based version of CTL described above. Consider the model of a system to be a labelled transition system $\mathcal{M}$. In addition, we can specify a constraint η (e.g., a fairness constraint) in $asCTL$ which is used to constrain the set of traces of $\mathcal{M}$ used by the verifier. Your model checker should allow us to check both $\mathcal{M} \models_\eta \varphi$ and $\mathcal{M} \models \varphi$ for a given $\mathcal{M}$, η and φ.

To help you with your implementation some components are provided for you in studres. You are given a model builder, a parser for $asCTL$, and an automation tool to generate test reports. There is also a readme file that explains in detail how the parser works, and how you need to specify the formulae.

A labelled transition system as covered in lectures is a tuple (S, Act, T, I, AP, L) where S is a finite set of states, Act is a finite set of actions, $T \subseteq S \times Act \times S$ is a set of transitions, $I \subseteq S$ denotes the initial states, AP is a finite set of atomic propositions and $L : S \to 2^{AP}$ is a labelling function which associates to each state a set of valid atomic propositions on that state.

In the implementation, a model is given as a `json` object where states and transitions are given explicitly, and AP and L implicitly by the information associated to states.

An $asCTL$ formula

$$\forall_A \mathcal{F}_B (p \wedge q)$$

with $A = \{act_1, act_2\}$ and $B = \{act_3, act_4\}$ is written in `json` as:

```
{
 "formula":  "AaFb (p && q)",
 "a":  ["act1", "act2"],
 "b":  ["act3", "act4"]
}
```

Both fairness constraints and properties that we want to check are written in the same way as a `formula`.

You are required to implement the method `check()` which takes as arguments a model, a fairness constraint (which can be just `True`) and a formula that we want to check. If this fails, you will need to return a trace (aka path) in the model that shows how the formula is not satisfied.

To show that your implementation works correctly you should provide a realistic model such as the mutual exclusion example treated in the lectures. You can use the model as given in the lectures, add fairness constraints and check different properties over that model. At least one property used should have actions.

Extensions

Extend your model checker by considering either of the following:

- Any additional features to improve your model checker.
- A different underlying model where we can express true concurrency. One such model can be a Petri net. Can you think of other properties you may want to express over true-concurrent models in a logic?
- Mechanisms for improving scalability and/or efficiency.

Submission

You should write a report discussing the semantics of $asCTL$ as supported by your model checker, assumed logic equivalences with justifications, and any design and implementation decisions of the model checker for the logic $asCTL$. Include evidence of testing and explain (the rationale of) extension activities done (if any). If you carried out an extension activity make it clear in the report what you have done. Please submit a single `.zip` archive containing all your code and your report as a `.pdf` to MMS by the specified deadline. As this practical is done in groups of 2, both students should submit the same information through their personal entry in MMS.

Lateness

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof).

Good Academic Practice

The University policy on Good Academic Practice applies.