LTL2AIG Benchmarks

Guillermo A. Pérez
Université Libre de Bruxelles – Brussels, Belgium
gperezme@ulb.ac.be

***Abstract*—We present a procedure to translate LTL synthesis specifications into the extended AIGER format for Synthesis. We give a simple example and take it through all the steps of the translation algorithm.**

I. Introduction

The original AIGER format [1] is a safety property specification used for model checking purposes. The extended version of it has been recently introduced by Jacobs [2] as a standard way of representing safety objectives for controller synthesis. The first use of this format is expected to be the Synthesis Competition 2014, a satellite event of the 26th International Conference on Computer Aided Verification 2014.

In [3] the authors present an algorithm to synthesize strategies for objectives specified using linear temporal logic (LTL) (see, e.g. [4]). Their algorithms take as input an LTL formula and a partition of the atomic propositions into controllable and uncontrollable signals. The idea of the algorithm is to 1) translate the negation of the LTL formula into a universal co-Büchi automaton, and 2) play a k-co-Büchi game on the extended subset construction of it. That is, the controller is allowed to visit configurations with accepting states at most k times. They increase this k until a winning strategy for the controller is found or k goes above a threshold (which is doubly-exponential in the size of the input formula). At this point the controller has a winning strategy in the constructed k-co-Büchi game if and only if she has a winning strategy in the original universal co-Büchi game.

II. LTL to AIGER

We have translated several benchmarks provided in the Acacia+ tool package [5], [6] into the extended AIGER format. The steps we follow closely resemble the algorithm presented in [3]. We take as input an LTL formula, a partition of the atomic propositions and a positive integer $k > 0$. Thus, if the AIGER safety objective specification is realizable, then the original LTL objective is also realizable (but the converse does not hold in general).

A. LTL formula to universal co-Büchi automata

We give a simple LTL specification which we will translate to the desired format in the sequel. The set of controllable atomic propositions is $\{grant\}$ while the uncontrollable ones are $\{req\}$.

$$\Box(req \rightarrow \Diamond grant); \tag{1}$$

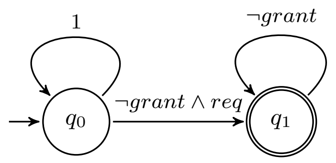

Fig. 1. Universal co-Büchi automaton for the negation of LTL formula 1.

We assume the reader has some familiarity with linear temporal logic. Intuitively, the above formula states that controller must eventually grant every request.

The negated formula is then translated into its corresponding universal co-Büchi automaton via, say ltl2ba [7], [8]. Figure 1 depicts the automaton for Equation 1.

B. From universal co-Büchi automata to AIG

As explained in the introduction, we will construct a game in which the controller has to avoid visiting accepting states more than k times. We achieve this by having $k+2$ copies of each state, one for each element from the sequence $0, \ldots, k+1$. Intuitively, we encode into the current state how many times the play has reached accepting states. This will be reflected in the new transition relation we define. If the play is in a copy of state q after having visited j accepting states and the next state visited, s, is accepting, then the actual state to which we transition is the $(j+1)$-th copy of s.

We now describe the translation in a formal fashion. Let F be an LTL formula with controllable and uncontrollable atomic proposition setst $\Sigma_u = \{req\}, \Sigma_c = \{grant\}$. Denote by Q the set of states from the automaton constructed from $\neg F$, $\Delta \subseteq Q \times \Sigma_u \times \Sigma_c \times Q$ its transition relation and $\mathcal{B} \subseteq Q$ the set of accepting states.

AIGER specifications are composed of a set of latches L, a set of inputs I, a single error function and a set of gates. Additionally, in the extended AIGER format one is expected to mark inputs as controllable if they are so. We take the set of latches L to be $|Q| \times \{0, \ldots, k+1\}$. For simplicity, we denote every latch by a state and number pair, e.g. $l_{(q_1,1)}$. For the special case of the initial configuration $(q_I, 0)$ we set

$$f_{l_{(q_I,0)}} = (\bigwedge_{l \in L} \neg l) \vee \bigvee_{(q,\sigma_u,\sigma_c,q_I) \in \Delta} l_{(q,0) \wedge \sigma_u \wedge \sigma_c}.$$

For every pair $s, i \in (Q \setminus (\mathcal{B} \cup \{q_I\})) \times \{0, \ldots, k+1\}$, we set the "next step" function of $l_{(s,i)}$ to be

$$f_{l_{(s,i)}} = \bigvee_{(q,\sigma_u,\sigma_c,s) \in \Delta} l_{(q,i)} \wedge \sigma_u \wedge \sigma_c$$

while for pairs $s, i \in (\mathcal{B} \setminus \{q_I\}) \times \{1, \ldots, k+1\}$ it is given by

$$f_{l_{(s,i)}} = \bigvee_{(q,\sigma_u,\sigma_c,s)\in\Delta} l_{(q,i-1)} \wedge \sigma_u \wedge \sigma_c.$$

The error function is then given by

$$f_{BAD} = \bigvee_{q\in Q} l_{(q,k+1)}.$$

We note the functions described above can be transformed into equivalent ones which only use logical conjunction and negation (required by the AIGER format). These, along with the input set $\Sigma_u \cup \Sigma_c$ already define the desired safety objective circuit.

C. Resulting AIG

We conclude by providing the resulting AIG file given by applying our algorithm to the automaton from Figure 1 when we fix $k = 1$.

```
aag 28 2 6 1 20
2
4
6 0
8 47
10 55
12 41
14 14
16 16
57
18 1 7
20 18 9
22 20 11
24 22 17
26 24 13
28 26 15
30 1 5
32 6 30
34 8 30
36 10 30
38 35 37
40 29 13
42 2 5
44 12 42
46 33 45
48 14 42
50 38 49
52 16 42
54 50 53
56 11 17
i0 req
i1 controllable_grant
l0 latch0
l1 latch1
l2 latch2
l3 latch3
l4 latch4
l5 latch5
o0 error
```

Latch 6 corresponds to the pair $(q_1, 0)$ and is therefore inaccessible. The initial configuration is latch 12. Indeed, it is 1 only if all latches are set to 0 or if latch 12 itself is set to 1. Observe the set of error configurations, as described by the circuit, is $\{(q_0, 2), (q_1, 2)\}$ – as expected.

References

[1] A. Biere. Aiger format and toolbox. [Online]. Available: http://fmv.jku.at/aiger/

[2] S. Jacobs. (2014, February) Extended aiger format for synthesis (v0.1). [Online]. Available: http://www.syntcomp.org/wp-content/uploads/2014/02/Format.pdf

[3] E. Filiot, N. Jin, and J.-F. Raskin, "Antichains and compositional algorithms for ltl synthesis," *Formal Methods in System Design*, vol. 39, no. 3, pp. 261–296, 2011.

[4] C. Baier, J.-P. Katoen *et al.*, *Principles of model checking*. MIT press Cambridge, 2008, vol. 26202649.

[5] A. Bohy, V. Bruyère, E. Filiot, N. Jin, and J.-F. Raskin, "Acacia+, a tool for ltl synthesis," in *Computer Aided Verification*. Springer, 2012, pp. 652–657.

[6] E. Filiot. Acacia+. [Online]. Available: http://lit2.ulb.ac.be/acaciaplus/

[7] P. Gastin and D. Oddoux, "Fast ltl to büchi automata translation," in *Computer Aided Verification*. Springer, 2001, pp. 53–65.

[8] P. Gastin. Ltl2ba: fast translation from ltl formulae to büchi automata. [Online]. Available: http://www.lsv.ens-cachan.fr/ gastin/ltl2ba