Bounded Δ-diagnosability of Timed Automata

Lulu He

March 2021

1 Encoding Bounded Δ-Diagnosability

Given: a TA with fault event F, observable and unobservable correct events; Δ, a positive rational (time spent in the faulty path after the first occurrence of F); bound, an integer (maximal length of the critical pair, i.e., common length of the faulty and normal paths, including the Nop transitions).
Input:
1. Parameters of the system
initial_state: integer, *bound*: integer, *delta*: positive_rational, *observable_events*: list_of_symbols, *fault*: symbol, *clocks*: list_of_symbols
The type *time_constraint* is defined as a finite set (with semantics of conjunct) of expressions of the form c^j op v_k where $c^j \in clocks$ for $1 \leq j \leq C$, $op \in \{<, \leq, >, \geq\}$ and v_k nonnegative rational.
We will also use a global clock gc, i.e., a clock which is never reset.
2. Transitions
Each transition $t = [q_s, e, q_d, g, r]$ has the following attributes:
source_state: integer, *event*: symbol, *destination_state*: integer, *guard*: time_constraint, *reset*: sub-list_of_clocks (0 if empty).
3. Invariants
To each state q is attached a state invariant $q.inv$, which is a time_constraint.
Read TA:
transitionList: stores all transitions in a finite list T. A transition is identified by its index i (positive integer) in the list and noted in short T_i for $transitionList[i]$.
$nextTransition[i]$: stores all next transitions of transition T_i, i.e., all those transitions whose source_state is equal to the destination_state of T_i.
Events are coded by integers: 0 for the Nop transitions, 1 for the fault F, 2 for all unobservable events and 3, 4, 5, ... for each observable event.
Initialization:
1. Add (at index 0) a Nop transition to transitionList, where:
Nop := $[-1, 0, -1, \{c^j \geq 0 \mid c^j \in clock\}, 0]$
It is actually a pattern of Nop transition as in the following, the fictional state -1 will be able to play the role of any state.
2. Add (at the end of the list) a fictional transition T_0 to transitionList that will play the role of the first transition of a timed path, with destination state the initial state, and will initialize the clocks at 0:
$T_0 = [maxState + 1, 2, initial\_state, \{c^j \geq 0 \mid c^j \in clock\}, \{c^j \mid c^j \in clock\}]$
where $maxState + 1$ is an artificial state.
3. Any transition, including Nop, can be followed by a Nop transition:
$nextTransition[\mathrm{i}].append(Nop)$, for all integer i.

For a given timed path, represented as a sequence $t_0, t_1, t_2, ..., t_n, n \leq bound$, of transitions $t_i = T_{k(i)}$, with $t_0 = T_{k(0)} = T_0$, in T (each one, possibly Nop, being in the *nextTransition* list of the previous one) occurring instantaneously at certain non-decreasing time instants, $delay[i], i \in [0, n]$ denotes the period between the occurrence of the event of t_i and that of t_{i+1} and represents a time transition of this period in the *destination_state* of t_i ($delay[n]$ is the period of stay in the *destination_state* of t_n up to the end of the timed path). A timed path is thus a sequence of alternating of discrete transitions and time transitions (possibly of period 0) beginning and finishing by time transitions (why we have added a fictional discrete transition T_0 at the beginning). A sequence of Nop transitions may occur between two discrete transitions, one can impose that the delay after a Nop transition is equal to 0 and thus the whole can be seen as time transitions of period 0 and do not change clock values and delays.

Values of clocks c^j and global clock gc when triggering transition t_i in the source_state of t_i, and at the end of the path after $delay[n]$ are noted $c^j[i]$ and $gc[i]$, $i \in [0, n+1]$. They are implemented as an array $c[1..C][0..n+1]$ and an array $gc[0..n+1]$ of nonnegative rationals, respectively.

All variables are duplicated for the faulty timed path and the normal timed path, with subscripts F and N respectively.

4. Initialize to zero the clocks in both paths:

$gc_F[0] = 0$
$gc_N[0] = 0$
$\{c^j_F[0] = 0 \mid c^j \in clocks\}$
$\{c^j_N[0] = 0 \mid c^j \in clocks\}$

Run:

```
for pos in range[0,bound]:
      s = Solver()
      s.add(Increase(pos))
      s.add(Constraints(pos))
      s.add(Delta_F[pos] = delta)
      res = s.check()
      if res = sat
            return model
            break
return unsat
```

Increase(pos): used to generate transitions $FaultyPath[pos]$ and $NormalPath[pos]$ in the faulty path and normal path respectively:

$FaultyPath[0] = T_0$ and, for $pos > 0$, $FaultyPath[pos] = t_{pos}, t_{pos} \in T \wedge t_{pos} \in nextTransition(k), T_k = lastActiveFaultyPath[pos-1]$

$NormalPath[0] = T_0$ and, for $pos > 0$, $NormalPath[pos] = t_{pos}, t_{pos} \in T \wedge t_{pos} \in nextTransition(l), T_l = lastActiveNormalPath[pos-1]$

Constraints(pos):

1. Last active (i.e., $\neq$ Nop) transition of faulty path and normal path (used because the $nextTransition$ of a Nop transition has for *source_state* the *destination_state* of the last active transition and not -1 which is a fictional state):

if $FautlyPath[pos] \neq Nop$
  $lastActiveFaultyPath[pos] = FautlyPath[pos]$
else
  $lastActiveFaultyPath[pos] = lastActiveFaultyPath[pos-1]$
if $NormalPath[pos] \neq Nop$
  $lastActiveNormalPath[pos] = NormalPath[pos]$
else
  $lastActiveNormalPath[pos] = lastActiveNormalPath[pos-1]$

2. Time progress during a discrete transition (which is instantaneous but possibly with a reset of some clocks) and during the subsequent delay:

for all $c^j \in clocks$:
  if $c^j \in FaultyPath[pos].reset$
    $c^{jr}_F[pos] = 0$
  else
    $c^{jr}_F[pos] = c^j_F[pos]$
  if $c^j \in NormalPath[pos].reset$
    $c^{jr}_N[pos] = 0$
  else
    $c^{jr}_N[pos] = c^j_N[pos]$
for all $c^j \in clocks$:
  $c^j_F[pos+1] = c^{jr}_F[pos] + delay_F[pos]$
  $c^j_N[pos+1] = c^{jr}_N[pos] + delay_N[pos]$
$gc_F[pos+1] = gc_F[pos] + delay_F[pos]$
$gc_N[pos+1] = gc_N[pos] + delay_N[pos]$

3. Satisfaction of the guard when triggering a discrete iransition:

$[[FaultyPath[pos].guard]](c_F[pos]) = True$

$[[NormalPath[pos].guard]](c_N[pos]) = True$

where $[[time\_constraint]](c[pos])$ means the evaluation of the time constraint for the values of clocks c^j given by $c^j[pos]$.

4. Satisfaction of the invariant of the *destination_state* of a transition after having triggered this transition and after the subsequent delay:

$[[FaultyPath[pos].destination\_state.inv]](c_F^r[pos]) = True$

$[[FaultyPath[pos].destination\_state.inv]](c_F[pos+1]) = True$

$[[NormalPath[pos].destination\_state.inv]](c_N^r[pos]) = True$

$[[NormalPath[pos].destination\_state.inv]](c_N[pos+1]) = True$

6. Fault occurrence:

$faultOccur_F[0] = 0$ and, for all $pos > 0$:

if $FaultyPath[pos].event = 1 \vee faultOccur_F[pos-1] = 1$

$faultOccur_F[pos] = 1$

else:

$faultOccur_F[pos] = 0$

$NormalPath[pos].event \neq 1$

7. Delta$_F$ (time spent after the first occurrence of the fault in the faulty path):

$Delta_F[0] = 0$ and, for all $pos > 0$:

if $faultOccur_F[pos] = 1$:

$Delta_F[pos] = Delta_F[pos-1] + delay[pos]$

else:

$Delta_F[pos] = 0$

8. No time elapsed after a Nop transition:

if $FaultyPath[pos].event = 0$:

$delay_F[pos] = 0$

if $NormalPath[pos].event = 0$:

$delay_N[pos] = 0$

9. Synchronization of timed observable events in faulty and normal paths:

if $FaultyPath[pos].event \geq 3 \vee NormalPath[pos].event \geq 3$:

$FaultyPath[pos].event = NormalPath[pos].event \wedge gc_F[pos] = gc_N[pos]$

10. Time period:

$delay_F[pos] \geq 0$

$delay_N[pos] \geq 0$

11. Breaking symmetries:

if $FaultyPath[pos] = Nop$:

$nopFaultyPath[pos] = 1$

else

$nopFaultyPath[pos] = 0$

if $NormalPath[pos] = Nop$:

$nopNormalPath[pos] = 1$

else

$nopNormalPath[pos] = 0$

$Or(Not(nopFaultyPath[pos]), Not((nopNormalPath[pos]))$

if $nopFaultyPath[pos] = 1$:

$Not(FaultyPath[pos+1].event \in \{1, 2\})$

if $nopNormalPath[pos] = 1$:

$Not(NormalPath[pos+1].event \in \{1, 2\})$