Tools and Algorithms for Deciding Timed Relations

Mihir Mehta
Department of Computer Science and Engineering,
Indian Institute of Technology, Delhi.
cs1090197@cse.iitd.ac.in

December 2012

Abstract

This is a report summarising the author's project on their B Tech Project for the academic year 2012-2013.

1 Objectives

- To develop a software toolkit that would enable users to verify various timed relations specifications and implementations expressed as timed automata.
  - To gain an understanding of the theory related to labeled transition systems, CCS processes and timed automata by surveying relevant literature.
  - To study tools already built by researchers for similar purposes.
  - To develop the software in a modular way with modules for language specification and modules for implementations of utility algorithms.
  - To implement algorithms for determining timed relations.

2 CCS processes

- A CCS process is an automaton with state and interfaces for interaction.
- The interaction is in the form of *actions* over communication ports known as *channels*.
- Given a port name a we refer to a as the label for input on the port and $\overline{a}$ as the label for the output on the port.

- *Inaction*: This is the simplest CCS process, denoted by 0. No state transitions or communcation can occur, in other words, this represents a deadlock.

- *Prefixing*: This is the simplest constructor; if P is a process and a is a label (input or output) then $a.P$ is also a process which can perform the action a in order to become the process P.

- *Naming*: We can give names to processes using syntax such as
  $Ndef$=$a_1.a_2$.....0
  This gives us the ability to define CCS processes recursively, such as this one:
  Parrot def→a.$\overline{a}$.Parrot

- *Choice*: If P and Q are processes, then $P + Q$ is a process as well which has the initial capabilities of both P and Q. The deadlock process 0 is the identity element for this, that is, $P + 0 = P$ is an identity.

- *Parallel Composition*: If P and Q are processes, then $P|Q$ is a process as well in which P and Q may proceed independently or communicate via complementary ports.

- *Restriction*: If P is a process and L is a set of channel names, then P/L is a process in which the component processes of P are the only processes which can communicate over channels from the set L.

- *Relabeling*: If P is a process and f is a function from labels to labels, then $P[f]$ is a process where each label from the domain of f is replaced by its image under f. One application of relabelling is the idea of *generic* processes: By relabelling the generic ports of such a process with specific port names, one can generate specific processes.

It is evident that each CCS process can be replaced by a labeled transition system (LTS) with equivalent behaviour, therefore we will, in the rest of this discussion, freely use the properties of LTS when describing those of CCS.

3 Equivalences on CCS

3.1 Trace equivalence

- A *trace* of an LTS is a sequence of actions that the LTS can perform.

- For an LTS P, the set $Traces(P)$ represents the set of all possible traces of P.

- Trace equivalence is said to exist between two LTS P and Q when $Traces(P) = Traces(Q)$.

- However, this notion proves to have a significant limitation in the case of CCS processes: Two CCS processes can have trace equivalence between their corresponding LTS and yet behave differently in terms of when they deadlock while interacting with a third CCS process.

3.2 Strong bismilarity

- Strong bisimulation: A binary relation R is a `strong bisimulation` if and only if, for all $(s_1, s_2) \epsilon R$ and $a \epsilon Act$.
$\forall s_1'(s_1 a s_1' \Rightarrow \exists s_2'.(s_2 a s_2' \wedge (s_1', s_2') \epsilon R)) \wedge$
$\forall s_2'(s_2 a s_2' \Rightarrow \exists s_1'.(s_1 a s_1' \wedge (s_1', s_2') \epsilon R))$

- It can be shown that the union of all strong bisimulations over the set of states is a strong bismulation. This binary relation is called `strong bisimilarity`, denoted by $\sim$.

- Better notion of equivalence than trace equivalence: picks up differences in the deadlock behaviour of processes under study.

- Failing: does not account for the invisible nature of τ transitions in CCS processes.

3.3 Weak bismilarity

- Weak bisimulation: A binary relation R is a `weak bisimulation` if and only if, for all $(s_1, s_2) \epsilon R$ and $a \epsilon Act$.
$\forall s_1'(s_1 a s_1' \Rightarrow \exists s_2'.(s_2 a \Rightarrow s_2' \wedge (s_1', s_2') \epsilon R)) \wedge$
$\forall s_2'(s_2 a s_2' \Rightarrow \exists s_1'.(s_1 a \Rightarrow s_1' \wedge (s_1', s_2') \epsilon R))$

- It can be shown that the union of all weak bisimulations over the set of states is a weak bismulation. This binary relation is called `weak bisimilarity`, denoted by $\approx$.

- Better suited to CCS processes, as it ignores τ transitions, thus disregarding hidden behaviour within a process.

3.4 Kanellakis and Smolka's algorithm

- This is a naive algorithm for determining the bisimilarity relation for the set of processes in a labelled transition system.

- This relies on the notion of a splitter.

- Let $\pi = \{B_0, ..., B_k\}, k \geq 0$ be a partition of the set of states Pr in a labeled transition system.

- A splitter for a block $B_i \epsilon \pi$ is a block $B_j \epsilon \pi$ such that for some action $a \epsilon Act$, some states in B_i have a-labelled transitions whose targets lie in B_j while other states in B_i do not.

- This suggests a refinement of π: replace block B_i with
  $B_i^1 = B_i \cap T_a^{-1}[B_j]$
  $B_i^2 = B_i - B_i^1$
- Refinements of this kind constitute the steps of this algorithm.
- The time complexity of this algorithm is $O(mn)$, since there can be at most n iterations, and all m edges are scanned in each iteration.

3.5 Fernandez' algorithm

- More efficient algorithm (O(m log n)).
- Relies on Paige and Tarjan's technique of three-way splitting.
- Splitters can now be 'simple' or 'compound'.
- Stability: A partition π is said to be stable with respect to a compound block S if S is not a splitter for any block in π for any action.
- For a compound block S, having a constituent simple block B satisfying $n(B) \leq 0.5 * n(S)$, and with respect to which π is stable, we can split a block B_i on an action a as follows:
  $B_i^1 = (B_i \cap T_a^{-1}[B]) - T_a^{-1}[S - B]$
  $B_i^2 = (B_i \cap T_a^{-1}[S - B]) - T_a^{-1}[B]$
  $B_i^3 = B_i \cap T_a^{-1}[B] \cap T_a^{-1}[S - B]$

4 Timed automata

- Formally, a timed automaton over a finite set of clocks C and a finite set of actions Act is a 4-tuple (L, l_0, E, I).
- L is a finite set of locations.
- l_0 is the initial location.
- $E \subseteq L \times B(C) \times Act \times 2^C \times L$ is a finite set of edges.
- $I : L \to B(C)$ assigns invariants to each edge location.
- $B(C)$ is the set of clock constraints over C. An element of $B(C)$ can be an equality, a slack inequality, a strict inequality, or an AND combination of such constraints.

5 Equivalences on Timed Automata

5.1 Timed bismilarity

5.2 Time abstracted bisimilarity

5.3 Regions and region graphs

5.4 Zones and zone graphs