Modeling and Verification of Time Critical Systems with the Fiacre/TINA toolbox

Bernard Berthomieu, Silvano Dal Zilio, François Vernadat
LAAS/CNRS, 7 avenue du Colonel Roche, 31077 Toulouse, France
Bernard.Berthomieu@laas.fr

AESE/STAE Forum Méthodes Formelles
Le Model-Checking en action
Toulouse – October 2014
Plan

Time Petri nets

The TINA toolbox

LTL model-checking

Fiacre

Projects
Automata
Automata as Petri nets
Graphs of tasks (e.g. PERT method)
Graphs of tasks as Petri nets
More features/idioms

Counters

Message passing

Mutual exclusion
Petri nets

\((P, T, \text{Pre}, \text{Post}, m_0)\) \hspace{1cm} (Petri 62)

state = marking, \(m, m', \ldots\)

t enabled at \(m\) iff \(m \geq \text{Pre}(t)\)

Reachability relation: \(m \xrightarrow{t} m' = m - \text{Pre}(t) + \text{Post}(t)\)

Properties

general: boundedness, liveness, absence of deadlocks

specific: reachability, LTL, CTL, etc (temporal logics)

Verification methods

marking graph, partial order, unfolding, structural, etc

Links: http://www.informatik.uni-hamburg.de/TGI/PetriNets
(Merlin 74)

\((P, T, \text{Pre}, \text{Post}, M_0, I_s)\), where

- \((P, T, \text{Pre}, \text{Post}, M_0)\) is a PN
- \(I_s : \text{Static Interval Function}\)
  
  \[ t \mapsto I_s(t) \subseteq R^+ \text{, rational bounds} \]
Behavior, states
simulation ...
simulation ...
simulation ...
simulation ...
simulation ...
simulation ...

[Diagram of a Petri net with places and transitions labeled with numbers, such as t5, t1, t2, t3, and p1, p2, p3, p4, p5. The diagram includes delays and initial markings.]
simulation ...
simulation ...
So

A state is:

A marking and a firing domain

Two kinds of transitions:

Delay transitions (time elapsing)

Discrete transitions (net transitions firing)

Delays may be:

Discrete: then state space if finite iff TPN is bounded

Dense: + expressive but state space generally infinite; abstractions needed
\[ E_0 = (m_0, I_0) \]
\[ m_0 : p_1, p_2(2) \]
\[ I_0 : \text{solutions in } t_1 \text{ of} \]
\[ 4 \leq t_1 \leq 9 \]
\[ E_0 \xrightarrow{\theta_1.t_1} E_1 = (m_1, I_1) \text{ with } (\theta_1 \in [4, 9]): \]
\[ m_1 : p_3, p_4, p_5 \]
\[ I_1 : \text{solutions in } (t_2, t_3, t_4, t_5) \text{ of} \]
\[ 0 \leq t_2 \leq 2 \]
\[ 1 \leq t_3 \leq 3 \]
\[ 0 \leq t_4 \leq 2 \]
\[ 0 \leq t_5 \leq 3 \]

\[ E_1 \xrightarrow{\theta_2.t_2} E_2 = (m_2, I_2) \text{ with } (\theta_2 \in [0, 2]): \]
\[ m_2 : p_2, p_3, p_5 \]
\[ I_2 : \text{solutions in } (t_3, t_4, t_5) \text{ of} \]
\[ \max(0, 1 - \theta_2) \leq t_3 \leq 3 - \theta_2 \]
\[ 0 \leq t_4 \leq 2 - \theta_2 \]
\[ 0 \leq t_5 \leq 3 - \theta_2 \]

The schedule \((5.t_1.0.t_2)\) is firable.
State classes

Parametric states

Given a firing sequence $\sigma$, one can compute system:

$$A(\Theta|\Phi) \leq b$$

where:

- $\Theta$ is vector of delays along sequence $\sigma$
- $\Phi$ describe the possible firing domains after $\sigma$

State classes

Observation: next firable transitions only depends on firing domain
State class after $\sigma$ = above system with $\Theta$ eliminated
Represented by a marking and a system of differences (DBM)
If the net is bounded, then there are finitely many state classes

Path systems

Above system with $\Phi$ eliminated.
Example

\[ C_0 = (p_1, p_2, \{4 \leq t_{19}\}) \]

\[ C_1 = (p_3, p_4, \{0 \leq t_2 \leq 4, 5 \leq t_3 \leq 6, 3 \leq t_4 \leq 6\}) \]

\[ C_2 = (p_2, p_3, \{1 \leq t_3 \leq 6, 0 \leq t_4 \leq 6, t_3 - t_4 \leq 3, t_4 - t_3 \leq 1\}) \]

\[ C_3 = (p_2, p_3, \{5 \leq t_3 \leq 6, 3 \leq t_4 \leq 6\}) \]

\[ C_4 = (p_3, p_4, \{0 \leq t_2 \leq 1, 5 \leq t_3 \leq 6, 3 \leq t_4 \leq 6\}) \]

\[ C_5 = (p_2, p_3, \{4 \leq t_3 \leq 6, 2 \leq t_4 \leq 6, t_3 - t_4 \leq 3, t_4 - t_3 \leq 1\}) \]
Using state classes

Several constructions, preserving either

1. markings
2. markings and complete traces
3. states
4. states and complete traces
5. branching properties

(2) or (4) adequate for LTL model checking

Check LTL property on time-abstracted behavior

Time counter example with path system

replay timed counter example in simulator
For added expressiveness

Priorities

Stopwatches – preemption

High level – colors or synchronized data processing
The TINA Toolbox (http://www.laas.fr/tina)

**nd (netdraw)**

- Graphic and textual editor
- Of Time Petri Net or Transition Systems
- Drawing, printing functions
- Interfaced with other tools

**tina, sift** (state space generators)

- Input: Petri nets + time, priorities, preemption, data (API)
- Builds: behavior abstractions, Preserving some classes of properties
- Output: in verbose form or in model-checkers formats

**selt, muse**

- Model checkers for State/Event-LTL temporal logic and modal mu-calculus

**struct, plan, etc**

- Structural analysis, Path analysis, converters, etc
State space generators (tina, sift)
TINA typical verification workflow

If fiacre input, compile first into tina “tts” description with frac

Build state space preserving the properties of interest (LTL or reachability) with tina or sift

Check properties on state space with selt or muse

If failed, compute a counter example with plan and analyze it in the simulator nd or play
Atomic propositions

on states

on transitions

boolean ($s34, t12, \text{idle}, \ldots$)

integer ($x \leq y + 2, s12 \geq 4.z, \ldots$)

Modal and Logic operators

Trace = alternating sequence of states and transitions

(For all traces)

$P$ P true at the first state (transition)
$\bigcirc P$ P true at the next state (transition)
$\square P$ P true at all states (transitions)
$\Diamond P$ P true at some state (transition)
$\square\Diamond P$ P true infinitely often
$\square(P \Rightarrow \Diamond Q)$ some state obeys Q after each state obeying P
Verifying $F$

Büchi Automata

$\omega$-automata

Let $F = \Box(P \Rightarrow \Diamond Q)$

Automaton associated with $\neg F$ (state 1 is accepting):

![Automaton Diagram]

Method

Build the synchronized product of behavior and Büchi Automaton of $\neg F$

counter example: path containing a circuit containing an accepting state
The SELT model checker

Formulas

S/E-LTL + arithmetics, e.g.

\( \Box(t1 \Rightarrow \Diamond(p2 \geq p3 + p4 \lor p6)) \)

Compacted counter examples

- [] (t1 => <> t4);
  FALSE
  state 0: p1 p2*2
  -t1 ... (preserving - t4 \& t1)->
  * [accepting] state 12: p3 p4 p5
  -t5 ... (preserving - t4)->
  state 12: p3 p4 p5

Counter examples can be replayed in the Tina simulator
Verification Example on TPN
Example ...
Example ...
Example ...
Example ...
Example ...
Example ...
Example …
Example ...
Example …
A higher level notation

Compiled to enriched TPN for tina

Rich language of properties (real-time patterns)
Features

Formally defined (mathematical semantics, compositional)

Types

Rich set of primitive data types; Strongly typed

Processes

Parameterized labelled automata

State transitions expressed symbolically; high-level constructions

Labels for communication and/or synchronization

May share variables with other components

Components

Hierarchically specified

Specify interactions between components or process instances

Constrain local interactions (time, priorities)

Control scope of shared variables
type index is 0..3
type request is union get_sum, get_value of index end
type data is array 4 of nat

process ATM [req : in request, resp : out nat] is
  states ready, send_sum, send_value
  var c : request, i : index, sum : nat, val : data := [6, 2, 7, 9]
  init to ready
  from ready
    req ?c;
    case c of
      get_sum -> to send_sum
      | get_value (i) -> to send_value
    end
  from send_value
    resp !val[i]; to ready
  from send_sum
    sum := 0;
    foreach i do sum := sum + val[i] end;
    resp !sum;
  to ready

component C [p : in nat] (&X : read nat) is
  port q : none in [2, 8]
  var Y : bool := false
  par p -> C1 [p,q] (X, Y)
  || p -> C2 [p,q] (X, Y)
end
Verification of Fiacre descriptions

Optimizing Compiler (frac):

Checks FIACRE descriptions

Optimizes/Abstracts

Translates into TINA format

Verification workflow similar to TPN

No graphic support yet ...
Verifying AADL descriptions

AADL = Architectural Analysis and Design Language

SAE standard (many companies involved)

Experiment:

AADL ⇒ FIACRE ⇒ TINA, and back
First arrow by model-transformations, second by compilation

Features supported:

Periodic/sporadic task activations
Fixed, non preemptive, scheduling
Communication by ports (immediate and delayed protocols)
Message queue prot.: OneItem, AllItems, DropOldest, DropNewest
AADL execution model (synchronous subset, async. extension)
Communications by shared variables
Modes
AADL2FIACRE

Translation scheme

One FIACRE process per AADL thread or device

Communication via a GLUE process managing:
  - Thread lifecycles
  - Scheduling
  - Communications (buffers, protocols)
  - Mode changes

Properties checked

- Schedulability
- Absence of deadlocks, deadlines preserved
- Non overflow of communication queues
- User properties, expressed in TINA/SELT logic
Projects

- LOTOS
- Front
- Flac
- Frac
- FIACRE
- TTS
- TINA
- CADP
- Polychrony
- SDL

AADL

- Spices
- QUARTESt
- CESAR
- openETCS

TOPCASED

OpenCmBeDD

aerospace valley
Links

Tina: http://www.laas.fr/tina
Fiacre: http://gforge.enseeiht.fr/projects/fiacre
Topcased: http://www.topcased.org
OpenEmbeDD: http://openembeddd.org
Spices: http://www.spices-itea.org
Quarteft: http://quarteft.loria.fr
Work in progress

Scaling up verification methods

symbolic methods (decision diagrams)

taking advantage of symmetries

compositional verification

parallel methods for model-checking

Fiacre tools – source level editing, simulation, verification

Specialized verification toolchains

AADL, SDL, SysML