DEPENDABILITY ENGINEERING WITH TIME-DEPENDENT PETRI NETS

("THE PROBLEM IS CHOICE")

CONTENTS

- motivation
- time-dependent Petri nets overview influence of time on qualitative properties zero test
- worst-case evaluation with duration interval nets counter example structural compression of well-formed net parts non-well-formed, but 1-bounded, acyclic, ... general procedure
- safety analysis with interval nets unreachableness of explicit error states example - concurrent pushers
PETRI NETS

MODEL

CLASSES

PLACE/TRANSITION

PETRI NET

(COLOURED PN)

context checking by
Petri net theory

verification by
temporal logics

TIME-DEPENDENT PN

TIME PETRI NET

worst-case
evaluation

STOCHASTIC

PETRI NET

performance
prediction

reliability
prediction

CONTINUOUS

PETRI NET

ODEs

WHICH KIND OF
TIME MODEL? (1)

- atomic sequential program parts -> transitions
  -> time assigned to transitions

- as simple as possible
  -> timed nets [Ramchandani 74]
  -> duration nets (D nets, DPN)

- duration nets
  -> constant times assigned to transitions
  -> token reservation
  -> firing consumes time

begin of
firing

end of
firing,
after a or b time units
Immediate Transitions

- zero (insignificant) time consumption
- time deadlocks \((-\rightarrow ZENONESS)\)
  - time deadlocks = state from which
    - no transient state is reachable
    - or: no state is reachable
      - where the system clock is able to advance
  - infinitely many firings in zero times
  - inconsistent time constraints!
  - How to avoid time deadlocks?
    - invariants?
    - \(\text{OPEN PROBLEM!}\)

How to analyse duration nets?

- time is running
  - change of the fire rule
    - \(pn\): t may fire \(\rightarrow\) t must fire single step \(\rightarrow\) maximal step
- special case:
  - duration of all transitions = 1 time unit
  - reachability graph construction under the maximal step firing rule
- else: transformation into special case
THE INFLUENCE OF TIME
EXAMPLE 1 (SYSTEM DEADLOCK)

Petri Net

different initial marking!

EXAMPLE 1
SYSTEM DEADLOCK,
MAX STEP RG = RG(DPN)

INA
ORD ROM NBM PUR CSV SCF CON SC Pt0 tF0 Fp0 pF0 MG SM FC EFC ES
Y Y Y Y N N Y Y N N N N N N N
DTP SMC SMD SMA CPI CTI B SB REV DSt BSt DTr DCF L LV L&
N Y Y N Y Y Y ? N N N N N

DSt (pn) -> not DSt (tpn)
EXAMPLE 1
SYSTEM DEADLOCK, REACHABILITY GRAPH

INIT STATE

DEAD STATE

RG (pn)
19 nodes, 32 arcs

RG (tpn)
6 nodes, 6 arcs

THE INFLUENCE OF TIME, EXAMPLE 2

not BND (pn) -> BND (tpn)
not DTr (pn) -> DTr (tpn)
not BND, simultaneously unbounded in m1 and m2

LIVE

BND,
- cycle time(p) = 2
- cycle time (s) = 2
- cycle time (c) = 1

not LIVE
- TSCC does not contain S_wait_m2
- S_wait_m2 is m0-dead
EXAMPLES, SUMMARY

- example 1
  - $\rightarrow \text{DSt (pn)} \rightarrow \text{not DSt (tpn)}$

- example 2
  - $\rightarrow \text{not BND (pn)} \rightarrow \text{BND (tpn)}$
  - $\rightarrow \text{not DTr (pn)} \rightarrow \text{DTr (tpn)}$

- generally

  ![Diagram: PN $\xrightarrow{T \rightarrow \text{TIME}}$ TPN]
  - $\text{prop(pn)} \xrightarrow{??} \text{prop(tpn)}$
  - $\text{RG (pn)} \supseteq \text{RG (tpn)}$

- BUT,
  for Petri net based system validation, we are only interested in the conclusions

  ![Diagram: prop(pn) $\xrightarrow{??} \text{prop(tpn)}$]

THE INFLUENCE OF TIME ON QUALITATIVE PROPERTIES

TIME-INSENSITIVE RESULTS

- $\text{BND (pn)} \rightarrow \text{BND (tpn)}$ ok
- $\text{not DSt (pn)} \rightarrow \text{not DSt (tpn)}$ ok
- $\text{DTr}_{m_0} (pn) \rightarrow \text{DTr}_{m_0} (tpn)$ ok

TIME-SENSITIVE RESULTS

- $\text{not BND (pn)} \rightarrow \text{BND (tpn)}$ ok
- $\text{DSt (pn)} \rightarrow \text{not DSt (tpn)}$ ok
- $\text{live (pn)} \rightarrow \text{not live (tpn)}$ ko ?
- $\text{REV (pn)} \rightarrow \text{not REV (TPN)}$ ko ?
- $\text{not REV (pn)} \rightarrow \text{REV (tpn)}$ ok

SUMMARY

- EF -properties: $\overline{\text{prop (pn)}} \rightarrow \overline{\text{prop (tpn)}}$
- AG EF-properties: $\text{prop (pn)} \leftarrow \text{prop (tpn)}$
### PROBE EFFECT

- **observation** - the system exhibits in test mode other (less) behaviour than in standard operation mode

- **cause** - sw test means (debugger) affect the timing behaviour

- **result** - masking of certain types of system behaviour / bugs
  - $DSt\ (pn) \rightarrow not\ DSt\ (tpn)$
  - $live\ (pn) \rightarrow not\ live\ (tpn)$
  - $not\ BND\ (pn) \rightarrow BND\ (tpn)$
  - $not\ REV\ (pn) \rightarrow REV\ (tpn)$

- **consequence** - systematic & exhaustive testing of concurrent systems is generally impossible

- **wayout** - qualitative models considering any timing behaviour

### TIME-ININVARIANT NET STRUCTURES

- **time-invariant == time independently live**

- **D nets** [Starke 90]
  - $\rightarrow$ homogeneous ES nets

<table>
<thead>
<tr>
<th>allowed</th>
<th>not allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **generalization ?**
  - $\rightarrow$ behavioural ES nets ?

- **troublemaker** - confusing combination of channel and control flow conflicts

<table>
<thead>
<tr>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>t2</td>
</tr>
<tr>
<td>t3</td>
<td></td>
</tr>
</tbody>
</table>

$\rightarrow$ “The problem is choice !”
CONFLUENCE

- concurrency and conflict overlap
  - case 1: $t_1 < t_3$
    - conflict $t_2 \# t_3$ disappears,
      firing of $t_3$ does not involve a conflict decision
  - case 2: $t_3 < t_1$
    - conflict $t_2 \# t_3$ exists,
      firing of $t_3$ involves a conflict decision

- the interleaving sequences of concurrency may encounter different amount of decisions
- an observer outside of the system does not know whether a decision took place or not

ARE THERE

TIME-INVARIANT

SOFTWARE STRUCTURES?
INFLUENCE OF COMMUNICATION PATTERNS ON NET STRUCTURE CLASSES

<table>
<thead>
<tr>
<th>addressing waiting \</th>
<th>direct / semi-direct-by-sender</th>
<th>indirect / semi-direct-by-receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic</td>
<td>EFC</td>
<td>ES</td>
</tr>
<tr>
<td>non-deterministic</td>
<td>ES</td>
<td>ICP</td>
</tr>
</tbody>
</table>

- simplified view
  -> provided, pre- and postprocesses do not access the same communication object from different control points

- known to be time-independently live [Starke 90]
  i.e. a live net remains live under any constant delay timing.

INFLUENCE OF COMMUNICATION PATTERNS ON CONFLICT STRUCTURES

<table>
<thead>
<tr>
<th>addressing waiting \</th>
<th>direct / semi-direct-by-sender</th>
<th>indirect / semi-direct-by-receive</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic</td>
<td></td>
<td>channel &amp; control flow conflicts appear only separately</td>
</tr>
<tr>
<td>non-deterministic</td>
<td>channel conflicts</td>
<td>confusing combination of channel &amp; control flow conflicts possible</td>
</tr>
</tbody>
</table>

- no dynamic
WHICH KIND OF TIME MODEL? (2)

- adequate characterization of time consumption
  - alternatives, iterations
  - time nets, [Merlin 74]
  - interval nets, I nets

- structural simplicity, e.g. alternative as
  - **duration net**
    - (with token reservation)
    - (constant times)
    - (firing consumes time)
    - working time
  - **interval net**
    - (no token reservation)
    - (interval times)
    - (firing itself timeless)
    - reaction time

- **duration interval net**, DI net
  - interval times
  - with token reservation
  - firing consumes time

NON-STOCHASTIC T-TIME-DEPENDENT PETRI NETS, OVERVIEW

<table>
<thead>
<tr>
<th>Firing principle</th>
<th>Working Time (token reservation)</th>
<th>Reaction Time (no token reservation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>timed nets [Ramchandani 74]</td>
<td>- / -</td>
</tr>
<tr>
<td></td>
<td>-&gt; (working time) duration nets</td>
<td>-&gt; reaction time duration nets</td>
</tr>
<tr>
<td></td>
<td><strong>D nets</strong></td>
<td></td>
</tr>
<tr>
<td>Interval</td>
<td>interval times</td>
<td>- / -</td>
</tr>
<tr>
<td></td>
<td>-&gt; working time</td>
<td>time nets [Merlin 74]</td>
</tr>
<tr>
<td></td>
<td>interval interval nets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-&gt; (reaction time) interval nets</td>
<td><strong>DI nets</strong></td>
</tr>
<tr>
<td></td>
<td><strong>I nets</strong></td>
<td></td>
</tr>
</tbody>
</table>
RELATION OF TIME-DEPENDENT PETRI NETS
(TRANSITION → TIME)

CONJECTURE !
A live Petri net remains live under any timing, [Popova 95]

- if it is persistent,
- if the earliest firing time of all transitions is zero
- if the latest firing time of all transitions is infinite
- if it is an homogeneous & timely homogeneous EFC without purely immediate transitions

A homogeneous & timely homogeneous EFC net is a net in which the transitions are enabled by the places and the firing times are within a specified interval.

Not homogeneous

Not timely homogeneous

- if it is an homogeneous & timely homogeneous behaviourally free choice net without purely immediate transitions.

\[ \text{pre}(t1) \cap \text{pre}(t2) \rightarrow AG (\text{enabled}(t1) \Leftrightarrow \text{enabled}(t2)) \]
WORST-CASE EVALUATION WITH DI NETS, INPUT PARAMETERS

- time consumption of sequential program parts
  - at least \( l \) time units
    (lower bound of duration time, \( \text{low}(t_{ij}) = l \))
  - at most \( m \) time units
    (upper bound of duration time, \( \text{upp}(t_{ij}) = m \))
  - or any (continuous) time in between measured by monitoring OR calculated from computer instructions

- no explicit branching probabilities

COMMUNICATION TIME MODEL

time to write into channel

sending

receiving

time to read from channel

transmission time (of communication medium)
WORST-CASE EVALUATION WITH DI NETS
OUTPUT PARAMETERS

- min execution duration (shortest path),
  max execution duration (largest path)

- esp. valuable for systems which require predictable timing behaviour (to meet given deadlines)

- calculations can be based on discrete reachability graph (only integer states)
  \[ t_{\text{begin, end}} = [\text{min}, \text{max}] \]

COMPUTATION OF MINIMAL PATH BY LOWER BOUNDS ONLY, COUNTER EXAMPLE

\[
\begin{align*}
\text{min\_duration(} & \text{dresden, weimar)}: \\
& \text{D net with lower bounds only: 14} \\
& \text{DI net with lower and upper bounds: 7} \\
& \rightarrow \text{maximal path by upper bounds only ? (!)}
\end{align*}
\]
COUNTER EXAMPLE AS D NET

dresden

dresden2leipzig

leipzig1

leipzig2

take_slow_train

take_fast_train

in_slow_train

in_fast_train

slow_train

fast_train

weimar

troublemaker

overlapping time windows of dresden2leipzig & berlin2leipzig

STRUCTURAL COMPRESSION OF WELL-FORMED NET STRUCTURES, EXAMPLE

init

par [2,3]

if

a1 [2,3]

a3 [1,5]

a4 [3,8]

eendif

loop

a6 [3,4]

a2 [3,5]

a5 [1,1]

p13

p23

endpar [1,2]

number of iterations
STRUCTURAL COMPRESSION OF WELL-FORMED NET STRUCTURES, EXAMPLE (CONT.)

1. init par [2,3]
   par [2,3]
   if
   p11
   a1 [2,3]
   p12
   a2 [3,5]
   p13
   a3+a4 [1,8]
   par
   [2,3]
   a6 [3,4]
   a7+endloop [0,20]
   p31
   p33
   endpar [1,2]
   a5 [1,1]
   loop
   endif

2. init
   par [2,3]
   if
   p11
   a1*a2 [5,8]
   a3+a4*a5 [2,9]
   par
   [2,3]
   a6*(a7+endloop) [3,24]
   p31
   p33
   endpar [1,2]
   a7+endloop [0,20]
   p31
   p33
   endpar [1,2]
   p23
   p33
   p31
   p33
   endpar [1,2]

3. init
   par [2,3]
   if
   a1*a2,(a3+a4)*a5 [5,24]
   a6*(a7+endloop),p23
   p33
   endpar [1,2]

4. init
   cycle [5,29]

5. \( t_{ik} = [a, b] \) \( t_{kj} = [c, d] \)

6. \( t_{ij} = [a + c, b + d] \)

7. \( t'_{ij} = [a, b] \)

8. \( t''_{ij} = [c, d] \)

9. \( t_{ij} = [\min(a, c), \max(b, d)] \)

10. \( t_{ij} = [m \cdot a + c, n \cdot b + c] \)

assumption:
\( \{ \text{lower} \}, \{ \text{upper} \} \) bound \( \{ \frac{m}{n} \} \) of iterations given
STRUCTURAL COMPRESSION OF WELL-FORMED NET STRUCTURES, GENERAL (CONT.)

\[
t_{ij'} = \left[ \max(\low(t_{i1j1}), \low(t_{i2j2})), \\max(\upp(t_{i1j1}), \upp(t_{i2j2}) \right] \\
t_{ij} = \left[ \low(t_{ij}) + \low(t_{ij'}) + \low(t_{jj}), \\upp(t_{ij}) + \upp(t_{ij'}) + \upp(t_{jj}) \right]
\]

EXAMPLE - INTERVAL EVALUATION OF NON-WELL-FORMED STRUCTURES, BUT 1-BOUNDED, ACYCLIC, ...

[Reske 95, p. 92]
INTERVAL EVALUATION, GENERAL PROCEDURE

- net structure transformation
  - [ first state -> init state ]
  - [ last state -> dead state ]
  - resolution of (unlimited) cycles, if any

- net type transformation
  - DI net -> I net or DI net -> D net;

- determine (set of) state numbers of
  - first state
  - last state
  - of the path to be measured;

- evaluation of
  - reachability graph OR ?
  - other descriptions of all possible behaviours
    - prefix of branching processes
    - concurrent automaton

EXAMPLE OF NET STRUCTURE TRANSFORMATION

MIN ( pathes(init, any dead state) );
MAX ( pathes(init, any dead state) );
ENVIRONMENT MODEL, WITH EXPLICIT ERROR STATES

PUSHER with error states

CONCURRENT PUSHERS, (PART OF THE) REACHABILITY GRAPH

init, R1_off, basic

Remark:
Only the interesting parts of the markings are shown.

-> (preemptive) interval nets

unreachability of bad states,
m₀-dead(ext2far) if:

\[ \text{lft}(tr2) < \text{eft}(\text{ext2far}) \land \text{lft}(\text{R1_set_off}) < \text{eft}(\text{ext2far}) \]
REFERENCES

[Bause 96]

[Heiner 97a]

[Heiner 97b]

[Merlin 74]

[Popova 91]
Popova, L.: On Time Petri Nets; J. Information Processing and Cybernetics EIK 27(91)4, 227-244.

[Popova 94]

[Popova 95]

[Ramchandani 74]

[Starke 90]
Starke, P. H.: Analysis of Petri Net Models (in German); B.G. Teubner 1990.

[Starke 95]