Weakly Bounded Petri Nets

Jörg Desel

KU Eichstätt-Ingolstadt

Evry, March 5th, 2010, séminaire MeFoSyLoMa
Weakly Bounded Petri Nets

Attention: Work in Progress!
Is this Petri net bounded?

No, the place s is unbounded!
Places s1 and s2 are unbounded

The place s1 is „worse unbounded“
weakly bounded

not weakly bounded
Why weakly bounded?

Analogy to weak liveness:
Why *weakly bounded*?

Analogy to *weak liveness*:

![Diagram of a Petri net](image_url)

*not live*
Why *weakly bounded*?

Analogy to *weak liveness*:

```

not live
```
Why *weakly bounded*?

Analogy to *weak liveness*:

weak liveness: *choices* can be controlled such that the controlled net behaves lively
Why weakly bounded?

Analogy to weak liveness:

weak boundedness: concurrency can be controlled such that the controlled net behaves boundedly.
Petri nets without branching places
Petri nets without branching places

Suggestion for a definition of weak boundedness:

• We are allowed to determine the (relative) speed of the components

For each occurrence sequence, we are allowed to change the order of concurrent transitions

Necessary requirement:

no component is (or becomes) inactive
i.e., we assume progress (weak liveness)
Petri nets without branching places

Progress assumption:

If transition $t$ is enabled then $t$ eventually occurs

An occurrence sequence will be called **progressing**, if it satisfies the progress assumption
Petri nets without branching places

Definition

A place $s$ is called **weakly $k$-bounded**
if each progressing occurrence sequence can be permuted
such that in the resulting occurrence sequence
$s$ carries never more than $k$ tokens.

A Petri net is called **weakly $k$-bounded**
if all its places are weakly $k$-bounded.

A Petri net is called **weakly bounded**
if each progressing occurrence sequence can be permuted
such that in the resulting occurrence sequence
only finitely many markings are reached.
Petri nets without branching places

Observation

**weak k-boundedness** implies **weak boundedness**
(if the set of places is finite)

**weak boundedness** implies **weak k-boundedness** for some k

This does not hold if further assumptions are made
for example:

- if the consumer is generally faster than the producer
  (talking about the average speed)
Example

Occurrence sequence:   b a b a c d b a b a c d b a b a c d ...
Permutation:           b a c d b a c d b a c d b a c d b a ...
Definition using partially ordered occurrence nets?
but ....
**Petri nets with branching places**

Idea (Cortadella, Kondratyev et. al.):

Petri nets model the control flow of concurrent programs which are executed sequentially (e.g. on one circuit).

The relative speed of the components can be controlled. Buffers are modelled by weakly bounded places.

Choices depend on (unknown) data. So choices can not be controlled.
weakly bounded
weakly bounded ???
Progress assumption:

If t is enabled then
either t occurs or a transition which is in conflict with t

An occurrence sequence is **progressing**
if it satisfies the progress assumption.
Fairness:

Each possible alternative will be selected eventually (each loop terminates …)

An occurrence sequence is called **fair** if it satisfies the fairness assumption.
weakly bounded !!!
Petri nets with branching places

Definition

A place $s$ is called **weakly $k$-bounded**, if each progressing **fair** occurrence sequence can be permuted where the order of alternatives (decision of choices) is kept such that in the resulting occurrence sequence $s$ carries never more than $k$ tokens.

A Petri net is called **weakly $k$-bounded** if all ist places are weakly $k$-bounded.

A Petri net is called **weakly bounded**, if each progressing **fair** occurrence sequence can be permuted where the order of alternatives (decision of choices) is kept such that in the resulting occurrence sequence only finitely many markings are reached.
A result

restricting assumptions:

- finitely many live state machines
- buffer places

a connected Petri net

the net (composed state machines + buffers) is live

choices are either free-choice (data dependent, if-then-else)
or controlled by buffer places (select statement)

Name: coupled state machines
A coupled state machine
state machines
Buffer places
data dependent choice
buffer controlled choice
The Result

A coupled state machine is weakly bounded

if and only if

The rank of ist incidence matrix equals \( |T| - |A\cdot I + |A| - 1 \)

where

- \( T \) – set of transitions
- \( A \) – set of free-choice alternatives
The result

Another formulation of

The rank of the incidence matrix equals $|T| - |A| + |A| - 1$:

$I$ Linearly independent $T$-invariants $I = 1 + \text{number of free-choice alternatives}$
1 T-invariant = 1 + 0 alternativs
0 T-invariants ≠ 1 + 0 alternativs
2 T-invariants  =  1 + 1 alternatives
3 T-invariants = 1 + 2 alternativs
2 T-Invariants $\neq 1 + 2$ alternatives
2 T-invariants = 1 + 1 alternativs

This Petri net is weakly bounded but not weakly k-bounded for any k
Schedulability Analysis of Petri Nets Based on Structural Properties

Cong Liu¹, Jörg Desel², Alex Kondratyev³, Yosinori Watanabe³, Alberto Sangiovanni-Vincentelli¹

¹University of California, Berkeley, USA
²Katholische Universität Eichstätt-Ingolstadt, Germany
³Cadence Berkeley Laboratories, USA

Scheduling Concurrent Programs

• The problem:

Given a set of concurrent non-terminating processes communicating through channels with infinite capacity, is there a sequential execution where channels are bounded?
Scheduling Concurrent Programs

• The problem:

Given a set of concurrent non-terminating processes communicating through channels with infinite capacity, is there a single process comprising the concurrent processes where channels are variables (arrays)?
Scheduling Classification

• Dynamic scheduling
  – Make all scheduling decisions at run-time
  – Context switch overhead

• Static scheduling [Lee 87]
  – Make all scheduling decisions at compile-time
  – Reduce context switch overhead
  – Restricted to specification without data-dependent controls (e.g. if-then-else)

• Quasi-static scheduling
  – Allow specification to have data-dependent controls
  – Perform static scheduling as much as possible
  – Leave data-dependent choices to be resolved at run-time
Quasi-Static Scheduling [Cortadella et al 00]

- Translate concurrent programs to a Petri net
- Find a quasi-static schedule for the Petri net
- Generate a sequential program from the schedule
Concurrent Programs → Petri Net

```
while(1){
    for(i=0,y=0;i<N;i++){
        read(CHAN, x, 2); y = y+x[0]+2*x[1];
    }
    write(OUT, y, 1);
}
```

```
while(1){
    read(IN, a, 1);
    b = a * a;
    write(CHAN, b, 1);
}
```

```
while(1){
    read(IN, a, 1);
    b = a * a;
    write(CHAN, b, 1);
} 
```
Petri Nets and Free Choice Sets

{C, D} is called a Free Choice Set (FCS).
It represents a data-dependant branch (if-then-else, loop)
We assume that each Free Choice Set has exactly two elements
Petri Nets and Free Choice Sets

Several input transitions?
(transitions with empty pre-set)
Petri Nets and Free Choice Sets

All input transitions (transitions with empty pre-set) generate a single Free Choice Set
Petri Nets and Free Choice Sets

All input transitions (transitions with empty pre-set) generate a single Free Choice Set.

We assume that each Free Choice Set has exactly two elements.
Schedule of a Petri net

– **finite** directed graph with a “root”
– **Vertices**: mapped to markings, root to initial marking
– **Edges**: transition occurrences, changing the marking
– Branching vertex: corresponds to a Free Choice Set
– strongly connected
Schedulability

A Petri net is schedulable if it has a schedule

Question: Is a given Petri net schedulable?
           Is a given Petri net not schedulable?

Solution 1: Try to construct a schedule
            very time consuming

Solution 2: Employ necessary conditions for schedulability
            which are based on the Petri net structure
            and hence efficient to decide.
            » Checking Cyclic Dependence of Transitions using
              Linear Programming
            » Checking a Rank Condition using Linear Algebra
Experiments

• Codecs
  – PVRG-JPEG encoder [Hung 93]
  – Motion-JPEG encoder [Lieverse 01]
  – Philips MPEG2 decoder [Wolf 99]
  – XviD MPEG4 encoder [Broekhof 04]

<table>
<thead>
<tr>
<th>Code</th>
<th>#P</th>
<th>#T</th>
<th>#Arc</th>
<th>#FCS</th>
<th>Rank</th>
<th>CDC</th>
<th>Scheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPEGenc1</td>
<td>26</td>
<td>27</td>
<td>64</td>
<td>6</td>
<td>&lt;0.01s</td>
<td>0.19s</td>
<td>&gt;24hr</td>
</tr>
<tr>
<td>MJPEGenc</td>
<td>117</td>
<td>124</td>
<td>330</td>
<td>25</td>
<td>&lt;0.01s</td>
<td>0.04s</td>
<td>&gt;24hr</td>
</tr>
<tr>
<td>MPEG2dec1</td>
<td>116</td>
<td>144</td>
<td>358</td>
<td>38</td>
<td>&lt;0.01s</td>
<td>0.25s</td>
<td>&gt;24hr</td>
</tr>
<tr>
<td>MPEG4dec</td>
<td>72</td>
<td>72</td>
<td>184</td>
<td>15</td>
<td>&lt;0.01s</td>
<td>0.16s</td>
<td>&gt;24hr</td>
</tr>
</tbody>
</table>
Related work

Weakly bounded Message Sequence Charts

(Anca Muscholl, Blaise Genest, Dietrich Kuske)
Open Questions

Decidability for the general case
   (idea: exclude loops in a coverability graph

Algorithms

Further interpretations (→ Monika Heiner)

Precise relation to weakly bounded MSCs

...