Petri nets in industrial applications

Jason Shamberger
Fabian Wenzel

EECS 249 Discussion
10/05/1999
Contents

1. Review of Petri nets
2. Modelling a batch plant
3. Deadlock problem
4. Conclusion, References
Part I: Review of Petri nets

- Visual and mathematical way of describing reactive systems
- Petri nets consist of arcs, places and (labeled) transitions
Review of Petri nets

- Ordinary PNs: All the arc weights are ones
- Bounded PNs: The number of tokens in each place is finite for any marking reachable from a specific initial marking
Concurrency in Petri nets

1. Interleaving of labeled PNs: Nets operate separately / asynchronously

\[
G_1 \parallel_{\Phi} G_2
\]
Concurrency in Petri nets

2. Full synchronization of labeled PNs:
Concurrency in Petri nets

3. Alternative connection:
Part II

Hierarchical Supervisory Control for Batch Processes
Modelling a batch plant

Valves (v1,v2,...)

Lines (l1,l2,...)

Supply tanks

Reactors

Storage tanks
Two products created by recipes
Possible Problems

- Deadlock between transporting devices
- Deadlock between processors
- Booking of all plant components becomes complicated (9 tanks + 16 valves $\rightarrow 2^{25}$ possible combinations)

Petri nets can be used to solve these problems
Hierarchical Supervisor

Original Plant

Valves

Lines

T

S

Abstraction
Models of Processors and Valves

Processor:

Valve:
Models of Connection Lines

\[ \{ u(T_1), u(T_2), u(T_3) \} \]
Joining / Splitting two Material Flows

- Typical phenomena of industrial processes
- Models of connection lines have to be modified with respect to the desired behavior
- Two possibilities for joining depending on the recipe:
  1. Synchronous
  2. Asynchronous
Booking models

- Describe booking of resources and transport devices on a higher level of abstraction
- Serve as the building blocks for recipes
- Make use of models of physical processors and lines
Booking Models of join and split

- Asynch. join
- Booking
- Unbooking of lines
- Freeing procs.
- Synch. join
- Split
Booking model of one recipe

\[ \langle \text{bp}_2, \{\text{bl}_9\}^{1}, \{\text{bl}_{10}\}^{2} \rangle \]

\[ \text{ul}_9 \]

\[ \{\text{bp}_6, \text{bl}_{11}\} \]

\[ \text{lu}_{11} \]

\[ \text{up}_2 \]

\[ \text{ul}_{10} \]
Part III: The deadlock problem

- Can occur both at lines and at processors, which are at different levels of abstraction
- Supervisors are implementations that take care of deadlock problems
- Different strategies can be applied
Deadlock at line level

No line can finish its action!
Solution: Supervisor T

- If n objects would lead to deadlock, work only with n-1, so that at least one can finish its action

- Algorithm:
  1. Examine the net topology
  2. If there are possible deadlock situations, create monitors for them
Algorithm Step 1

Examination of the Petri net topology

Core loops indicate possible deadlocks
Algorithm Step 2

Build monitors
Deadlock at the top-level

- Recipes change with time, so behavior is not static
- Booking of processors depends directly on recipes and therefore can still lead to deadlocks
Solution: Supervisor S

- Generate plant behavior by
  1. Interleaving Recipes
  2. Synchronous composition of the plant’s physical and logical components: Processors and Supervisor T (including lines)
  3. Removing remaining deadlock situations
- Can take care of optimization constraints, e.g. time, costs
Again: Hierarchical Supervisor

Original Plant

Abstraction

S → T → Lines → Valves → Procs.
Conclusion

- Petri nets can model industrial control processes and can be used to avoid deadlocks
- Hierarchical solution separates the plant from control components
- Number of states is reduced
References

- K. Åkesson and M. Fabian: Implementing supervisory Control for Chemical Batch Processes