Modeling Safety-Critical Systems with Z and Petri Nets

Monika Heiner[1] and Maritta Heisel[2]

[1] Brandenburgische Technische Universität Cottbus, Fachbereich Informatik, D-03013 Cottbus, email: mh@informatik.tu-cottbus.de
[2] Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, Institut für Verteilte Systeme, D-39016 Magdeburg, Germany, email: heisel@cs.uni-magdeburg.de

1 Introduction

```
Safety-critical systems have data-oriented as well as behavioral aspects.
Combination of languages ...What are advantages of this particular combination?
(Animation, formality, ...)
```

2 Modeling Principles

Each transition of a Petri net corresponds to an operation specified in Z. The goal of the combination is to obtain simple Petri nets. Data aspects need not be encoded in the nets, but can be specified in Z.

If a Z operation makes a case distinction, the two different transitions correspond to it.

To ensure compatibility of the two specifications, we have the following proof obligations:

- The initial marking of the Petri net must be consistent with the Z specification.
- The conditions associated with incoming places of transitions correspond to preconditions of Z operations, the conditions associated with outgoing places of transitions correspond to postconditions established by Z operations. Hence, for chains we have the obligation to show that the precondition of a later operation in the chain must be compatible with the postcondition established by the preceeding operation in the chain.
- For operations op_1 and op_2 where the Petri net admits concurrent execution, we must show that
  - the operations do not exclude each other, i.e., $\neg$ (pre $op_1 \wedge$ pre $op_2 \Leftrightarrow$ *false*)
  - for all states where pre $op_1 \wedge$ pre op_2 holds the order in which the operations are executed is irrelevant. Hence, our semantics of concurrency is interleaving of the corresponding Z operations.

Z operations are used to resolve conflicts in Petri nets. The petri net can engage in more behaviors than permitted by the Z specification. For safety-related properties, this does not seem to be a problem, because there we show that certain things cannot happen. Hence, if the net with the more liberal behavior is safe, than the more restricted behavior is also safe.

`Model of environment also expressed as Petri net, e.g., sensors`

3 Case Study: Production Cell

3.1 Z Part of the Specification

The specification follows the usual Z style. We begin with global definitions, followed by the internal state of the system. Finally, we present the system operations. Readers not familiar with Z are referred to [Spi92].

Global Definitions

$$YesNo ::= yes \mid no$$

$$OnOff ::= on \mid off$$

$$maxplates : {}_1$$

$$[Table\_Position]$$

$$load\_position, unload\_position : Table\_Position$$

$$
\begin{array}{|l}
next, prev : Table\_Position \quad Table\_Position \\
\_ < \_ : Table\_Position \quad Table\_Position \\
\hline
\operatorname{dom} next = Table\_Position \setminus \{unload\_position\} \\
\operatorname{dom} prev = Table\_Position \setminus \{load\_position\} \\
\forall tb : Table\_Position \mid tb \in \operatorname{dom} next \bullet tb < next(tb) \\
load\_position < unload\_position
\end{array}
$$

System State

$$
\begin{array}{|l}
\underline{\ feed\_belt\ } \\
fb\_mvt : OnOff \\
at\_front, at\_end : 0\,..\,1 \\
in\_between, number\_of\_plates : 0\,..\,maxplates \\
\hline
number\_of\_plates = at\_front + in\_between + at\_end \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{Init\_feed\_belt} \\
\textit{feed\_belt}' \\
\hline
\textit{number\_of\_plates}' = 0 \\
\textit{fb\_mvt}' = \textit{off} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{table} \\
\textit{t\_position} : \textit{Table\_Position} \\
\textit{t\_loaded} : \textit{YesNo} \\
\textit{t\_mvt} : \textit{OnOff} \\
\textit{can\_receive} : \textit{YesNo} \\
\hline
\textit{can\_receive} = \textit{yes} \\
\quad \Leftrightarrow \textit{t\_position} = \textit{load\_position} \land \textit{t\_loaded} = \textit{no} \land \textit{t\_mvt} = \textit{off} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{Init\_table} \\
\textit{table}' \\
\hline
\textit{t\_loaded}' = \textit{no} \\
\textit{t\_mvt}' = \textit{off} \\
\hline
\end{array}
$$

Operations

Table control operations

$$
\begin{array}{|l}
\hline
\textit{start\_unload\_to\_load} \\
\Delta \textit{table} \\
\hline
\textit{t\_loaded} = \textit{no} \\
\textit{t\_position} = \textit{unload\_position} \\
\textit{t\_mvt} = \textit{off} \\
\textit{t\_loaded}' = \textit{no} \\
\textit{t\_position}' = \textit{unload\_position} \\
\textit{t\_mvt}' = \textit{on} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{move\_unload\_to\_load} \\
\Delta \textit{table} \\
\hline
\textit{t\_loaded} = \textit{no} \\
\textit{load\_position} < \textit{t\_position} \\
\textit{t\_mvt} = \textit{on} \\
\textit{t\_loaded}' = \textit{no} \\
\textit{t\_position}' = \textit{prev}(\textit{t\_position}) \\
\textit{t\_mvt}' = \textit{on} \\
\hline
\end{array}
$$

stop_at_load

$\Delta table$

$t\_loaded = no$
$t\_position = load\_position$
$t\_mvt = on$

$t\_loaded' = no$
$t\_position' = load\_position$
$t\_mvt' = off$

start_load_to_unload

$\Delta table$

$t\_loaded = yes$
$t\_position = load\_position$
$t\_mvt = off$

$t\_loaded' = yes$
$t\_position' = load\_position$
$t\_mvt' = on$

move_load_to_unload

$\Delta table$

$t\_loaded = yes$
$t\_position < unload\_position$
$t\_mvt = on$

$t\_loaded' = yes$
$t\_position' = next(t\_position)$
$t\_mvt' = on$

stop_at_unload

$\Delta table$

$t\_loaded = yes$
$t\_position = unload\_position$
$t\_mvt = on$

$t\_loaded' = yes$
$t\_position' = unload\_position$
$t\_mvt' = off$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{unload\_table}} \\
\Delta table \\
\hline
t\_loaded = yes \\
t\_position = unload\_position \\
t\_mvt = off \\
\\
t\_loaded' = no \\
t\_position' = unload\_position \\
t\_mvt' = off \\
\hline
\end{array}
$$

Operations related to the feed belt environment

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{load\_fb}} \\
\Delta feed\_belt \\
\hline
number\_of\_plates < maxplates \\
at\_front = 0 \\
\\
at\_front' = 1 \\
in\_between' = in\_between \\
at\_end' = at\_end \\
fb\_mvt' = fb\_mvt \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{move}} \\
\Delta feed\_belt \\
\hline
fb\_mvt = on \\
at\_front = 1 \\
\\
in\_between' = in\_between + 1 \\
at\_front' = 0 \\
at\_end' = at\_end \\
fb\_mvt' = fb\_mvt \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{detect}} \\
\Delta feed\_belt \\
\hline
fb\_mvt = on \\
at\_end = 0 \\
in\_between > 0 \\
\\
at\_end' = 1 \\
in\_between' = in\_between - 1 \\
at\_front' = at\_front \\
fb\_mvt' = on \\
\hline
\end{array}
$$

Feedbelt control cperations

$$
\begin{array}{|l}
\hline
\textit{switch\_on} \\
\Delta \textit{feed\_belt} \\
\Xi \textit{table} \\
\hline
\textit{fb\_mvt} = \textit{off} \\
\textit{number\_of\_plates} > 0 \\
\textit{can\_receive} = \textit{yes} \lor \textit{at\_end} = 0 \\
\textit{fb\_mvt}' = \textit{on} \\
\textit{in\_between}' = \textit{in\_between} \\
\textit{at\_front}' = \textit{at\_front} \\
\textit{at\_end}' = \textit{at\_end} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{switch\_off} \\
\Delta \textit{feed\_belt} \\
\Xi \textit{table} \\
\hline
\textit{fb\_mvt} = \textit{on} \\
\textit{at\_end} = 1 \\
\textit{can\_receive} = \textit{no} \\
\textit{fb\_mvt}' = \textit{off} \\
\textit{in\_between}' = \textit{in\_between} \\
\textit{at\_front}' = \textit{at\_front} \\
\textit{at\_end}' = \textit{at\_end} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{fb\_to\_table} \\
\Delta \textit{feed\_belt} \\
\Delta \textit{table} \\
\hline
\textit{fb\_mvt} = \textit{on} \\
\textit{at\_end} = 1 \\
\textit{can\_receive} = \textit{yes} \\
\textit{at\_end}' = 0 \\
\textit{in\_between}' = \textit{in\_between} \\
\textit{at\_front}' = \textit{at\_front} \\
\textit{fb\_mvt}' = \textit{fb\_mvt} \\
\textit{t\_loaded}' = \textit{yes} \\
\textit{t\_position}' = \textit{t\_position} \\
\textit{t\_mvt}' = \textit{t\_mvt} \\
\hline
\end{array}
$$

3.2 The Petri Net Part of the Specification

Figure 1 shows the Petri net that specifies the order in which the various Z operations can be executed.

We can identify the following concurrent operations:

- The operation *load_feed_belt* is concurrent with *detect*, *switch_on*, *switch_off*, and *from_fb_to_table*.
- The operation *move* is concurrent with *detect*, and *from_fb_to_table*.

3.3 Validation

With our modeling, we should be able to demonstrate the properties mentioned in Section 2.3 of the LNCS book that concern the feed belt and the table, in particular:

- Blanks do not fall off the feed belt. The feed belt is stopped before this can happen. Hence, we must show that if $at\_end = 1 \wedge fb\_mvt = on \wedge can\_recieve = no$ then $fb\_mvt = off$ must hold in the next or the state after the next one (due to concurrency with the *load_feed_belt* operation).
- The blanks have sufficient distance so that they can be distinguished.
- The table does not move beyond its extreme points.

To show the first property, we need to analyze the Petri net. For the other two, I don't know.

4 Related Work and Conclusions

References

[Spi92] J. M. Spivey. *The Z Notation – A Reference Manual.* Prentice Hall, 2nd edition, 1992.

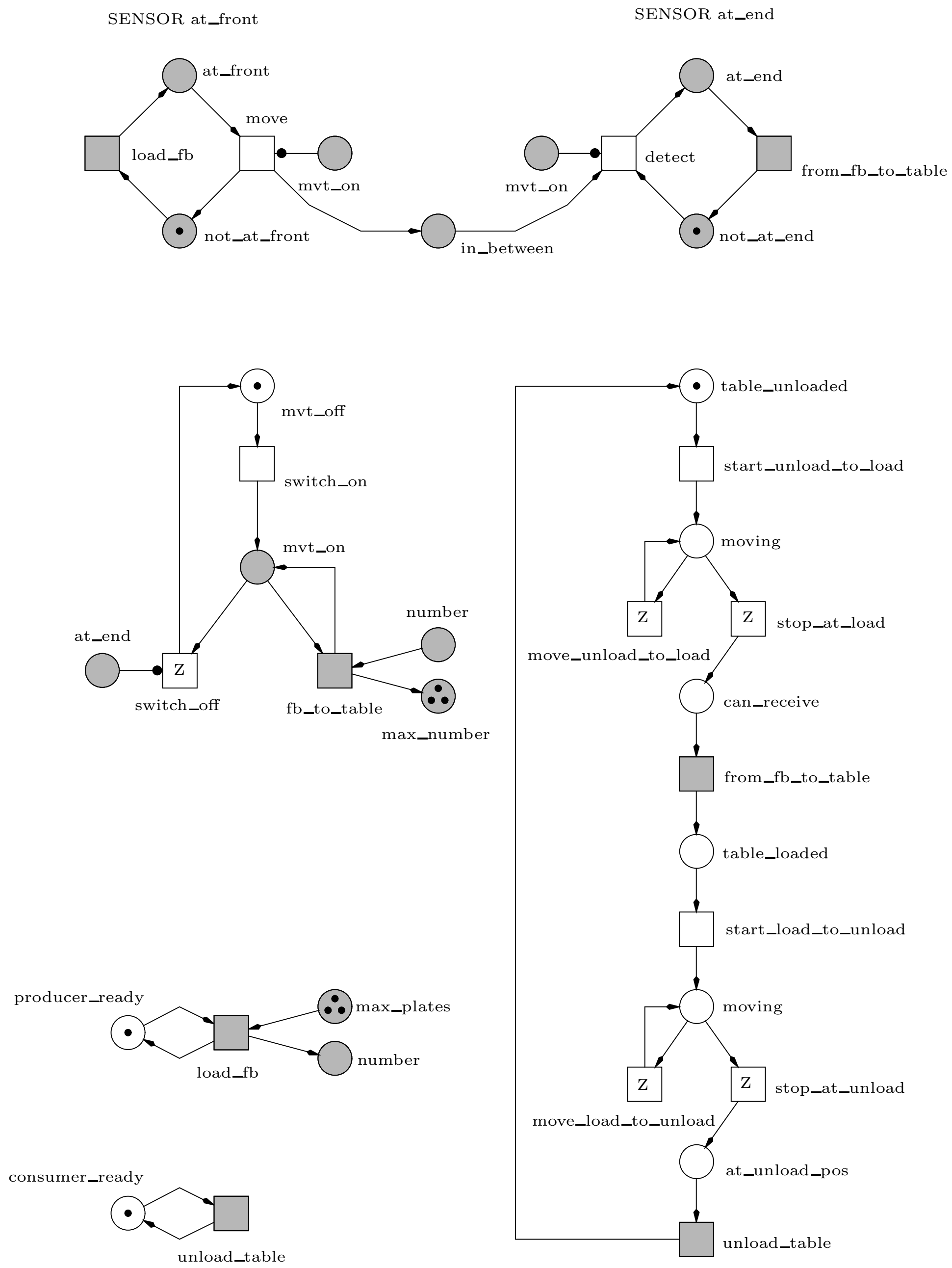

Fig. 1. Petri net for production cell