PETRI NET BASED DESIGN AND ANALYSIS OF LOGIC CONTROLLERS

Georg Frey

NSF Workshop on Logic Control for Manufacturing
University of Michigan, Ann Arbor, MI, June 26-27, 2000

Outline

- Motivation / Viewpoint
- Signal Interpreted Petri Net (SIPN)
- Correctness Analysis
- Transparency Analysis
- Implementation

UNIVERSITY of KAISERSLAUTERN
Institute of Process Automation
Prof. Dr.-Ing. habil. Lothar Litz
AT + Uni KL

MOTIVATION

Background: Challenges in industrial Controller Design

- High Complexity
- Reuse and Modification of Software
- More or less unique specification for each controller

Objective: Application of formal Methods

- Transparency and Intuition
- Use of Control Theory and Software Engineering Concepts

Limit: Formal Area

- “In this game we’re playing, we can’t win. Some kinds of failure are better than other kinds, that's all” George Orwell, 1984

VARYING VIEWS AND COMBINED VIEW

Control Theory

- SIPN as formalization of the informal problem description
- Question: Is the formal specification complete and consistent
- Implementational aspects are of interest (IEC 1131-3)

Software Engineering

- SIPN Control Algorithm as Application Software (ANSI/IEEE 610)
- Question: Software Quality (ISO/IEC 9126)
- Implementational aspects are of minor interest

Control Theory + Software Engineering

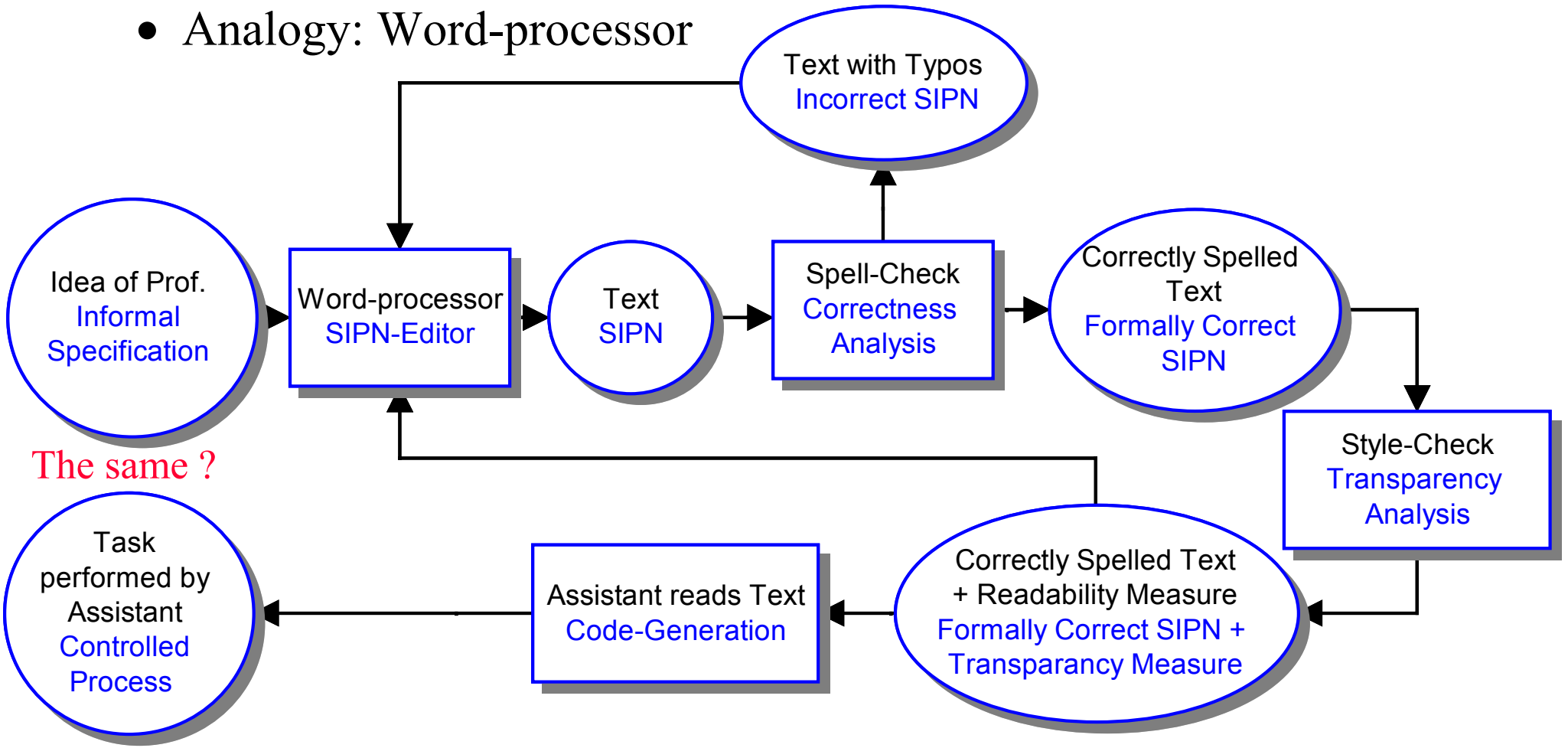
- SIPN as formal problem description AND as application software
- Question: Completeness, Consistency, AND Quality (Transparency)
- Implementational aspects are of interest (IEC 1131-3)

Pre-Condition: Implemented Software follows Formal Specification exactly

LIMIT

Formal Analysis can NEVER answer the question of SENSE

- Analogy: Word-processor

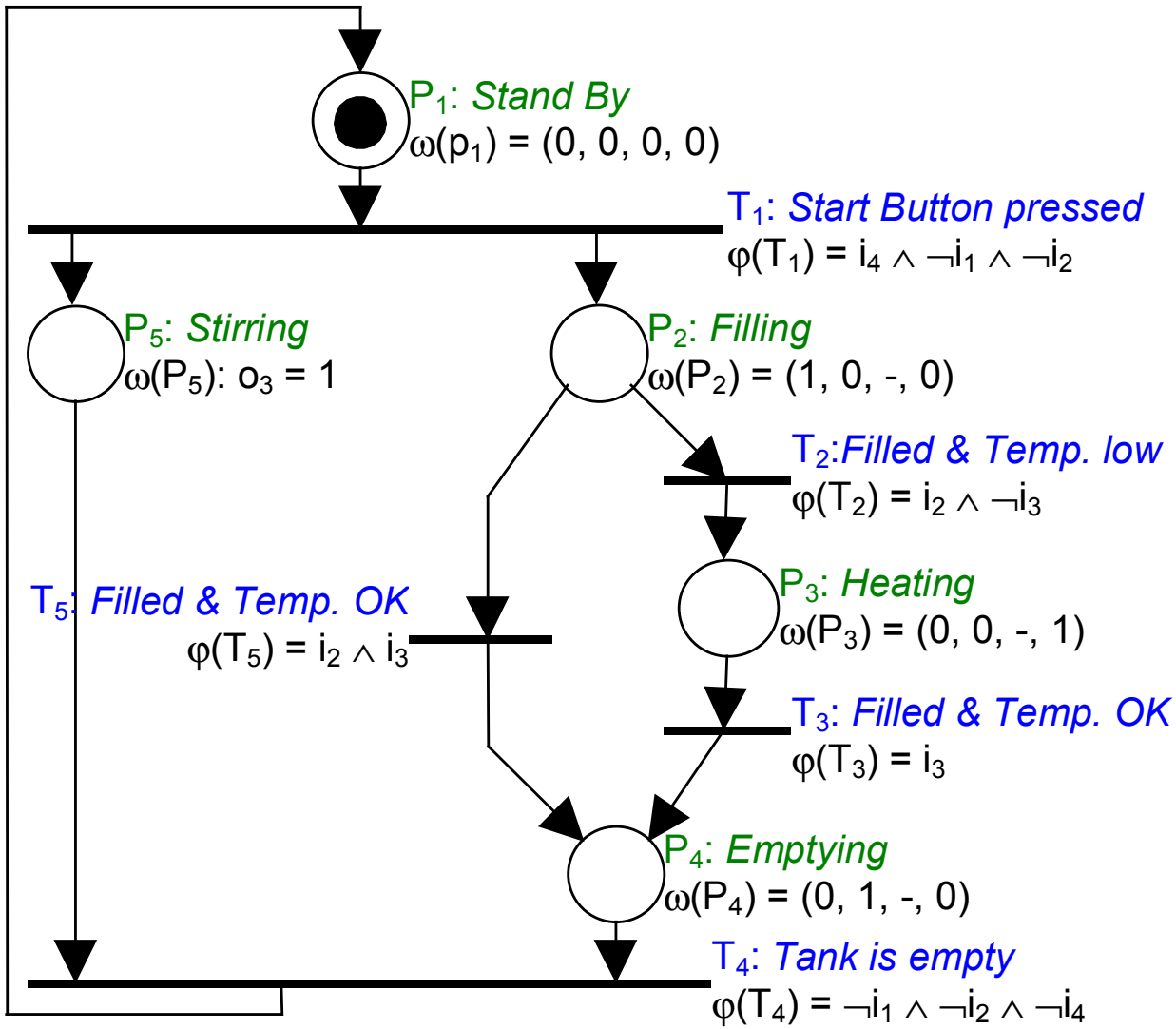


SIPN Based Control Algorithms

Signal Interpreted Petri Net (SIPN)

[Frey and Litz]

- Functions of Input Signal at Transitions
- Output Signals at Places
- Forced firing
- Synchronous firing
- Iterative firing



DYNAMIC SYNCHRONIZATION (DS)

Definition

Two transitions t_1 and t_2 form a dynamic synchronization if the firing of t_1 implies the simultaneous firing of t_2.

Classification

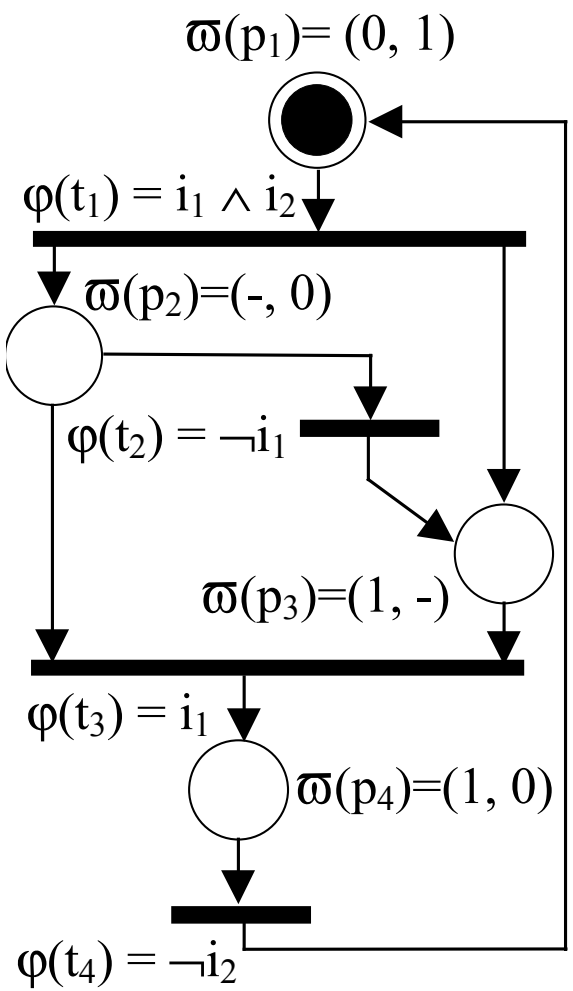
- Full (always synchronized)
- Partial (synchronized under special constraints)

Validity of PN Analysis for SIPN under full DS

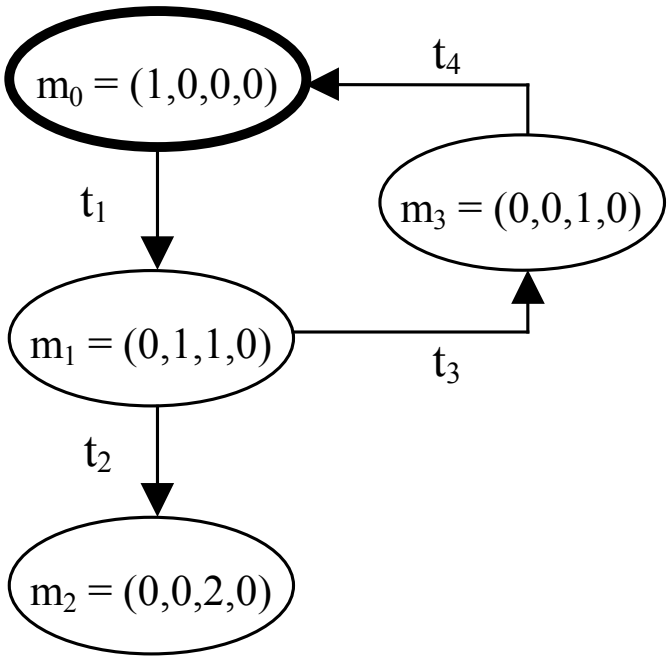
- Safety of the underlying PN is sufficient for the safety of the SIPN but not necessary.
- Liveness of the underlying PN is necessary but not sufficient for the liveness of the SIPN.
- Reversibility is neither necessary nor sufficient.

EXAMPLE 1: SAFETY & REVERSIBILITY

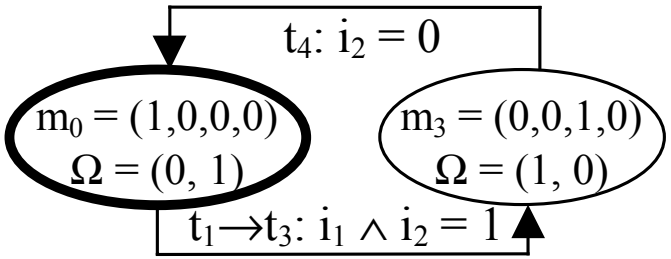
a) SIPN



b) RG_{PN}



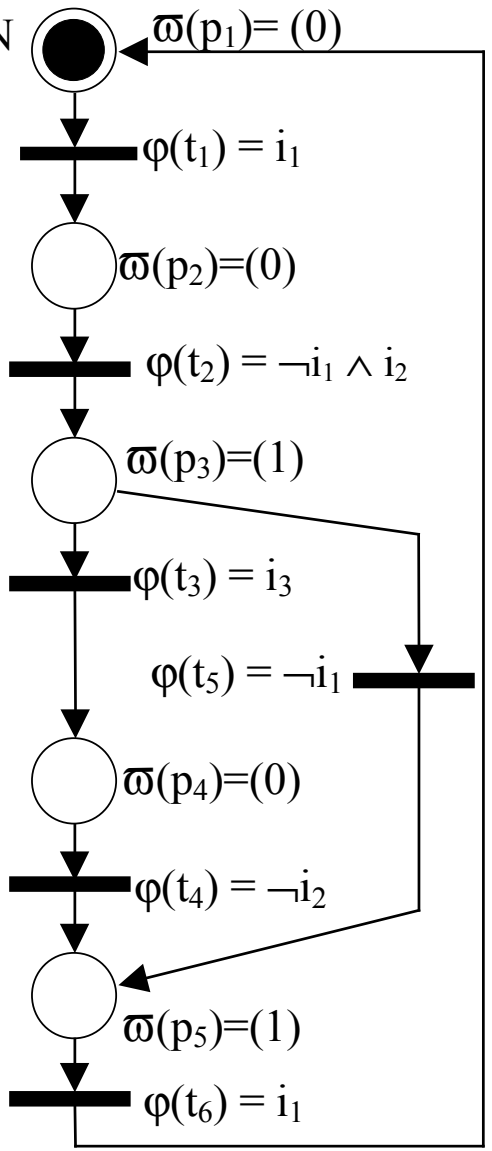
c) RG_{SIPN}



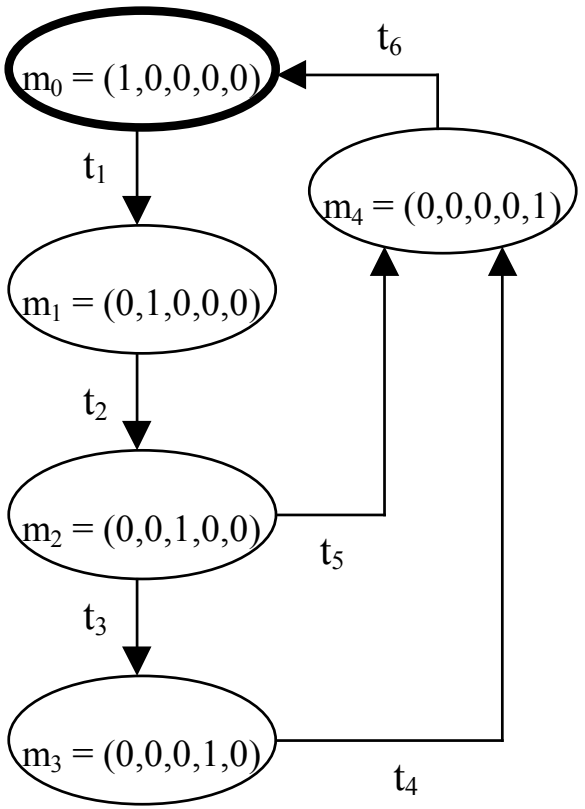
- **PN is not safe SIPN is (not sufficient)**
- **PN is not reversible SIPN is (not necessary)**
- **Both are not live**

EXAMPLE 2: LIVENESS & REVERSIBILITY

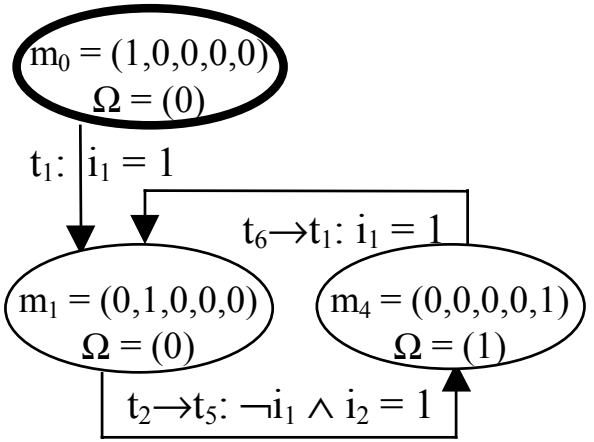
a) SIPN



b) RG_{PN}



c) RG_{SIPN}



- **PN is live SIPN not (not sufficient)**
- **PN is reversible SIPN not (not sufficient)**

Criteria for Correctness

Unambiguity

Every control algorithm has to be defined unambiguously. This criterion can be subdivided into four sub-criteria:

- Determinism
- Termination
- Defined output
- Unambiguous output

Liveness

When a transition or a set of transitions is not live then part of the control algorithm doesn't work anymore.

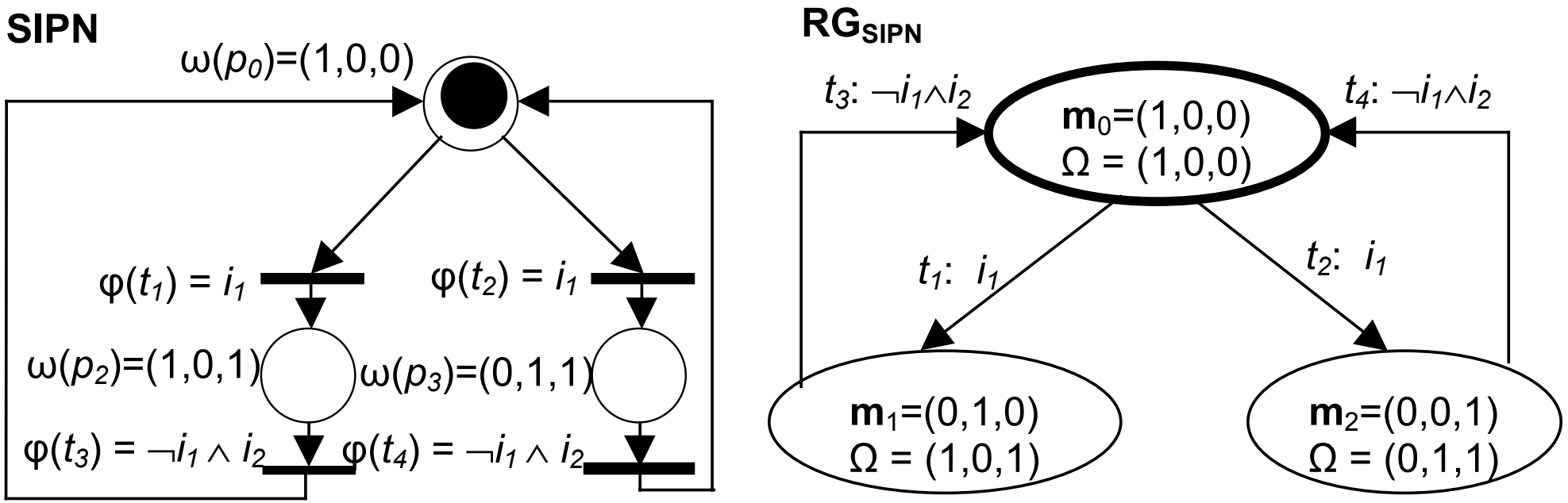
Reversibility

Reversibility guarantees that the described controller reaches its initial state again.

CRITERION 1a: DETERMINISM

Determinism: A control algorithm has to be deterministic. If it was not, its behavior in a given situation would depend on implementational aspects.

→ *The algorithm is deterministic if the firing conditions at every branching in RG_{SIPN} are disjoint.*

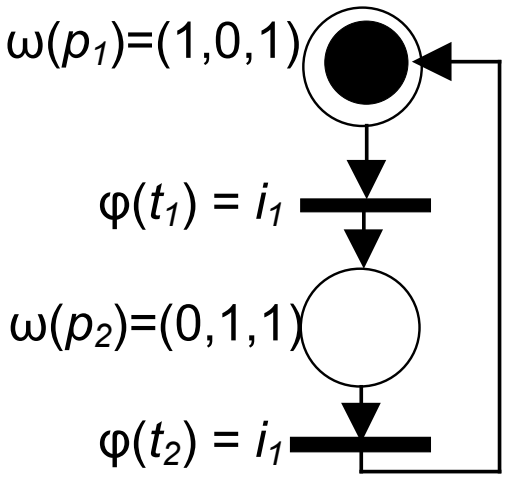


CRITERION 1b: TERMINATION

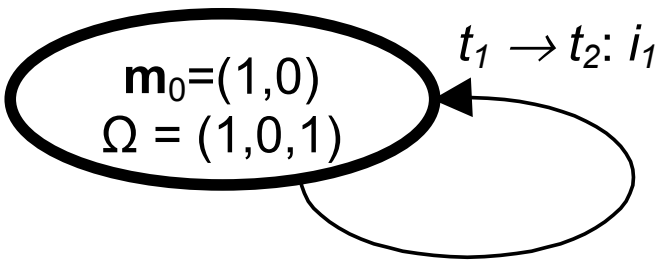
Termination: In a cycle of a logic control algorithm, at least one marking must be stable. A cycle without stable marking leads to an algorithm that does not terminate.

→ *The algorithm terminates if there is no self-loop at any state in RG_{SIPN}.*

SIPN



RG_{SIPN}

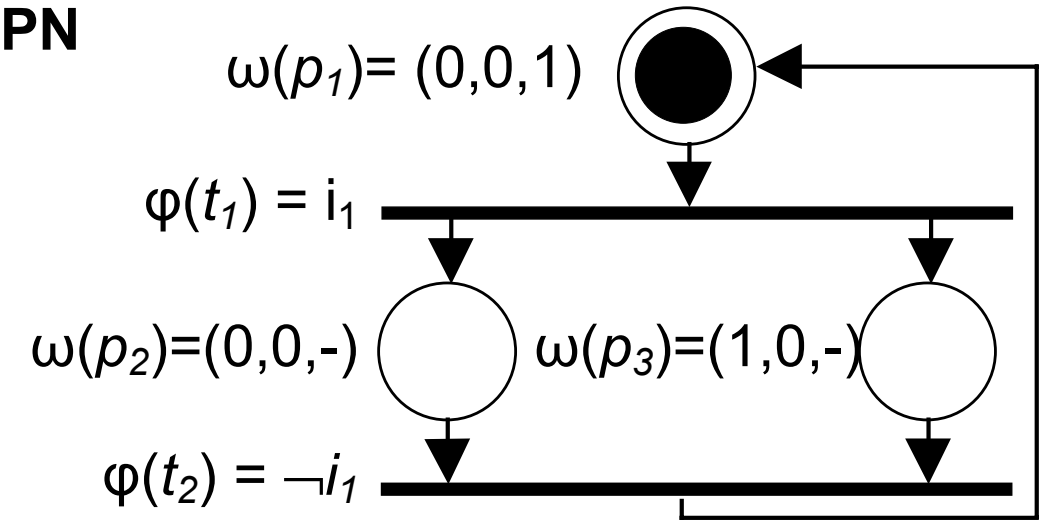


CRITERION 1c and 1d: OUTPUT

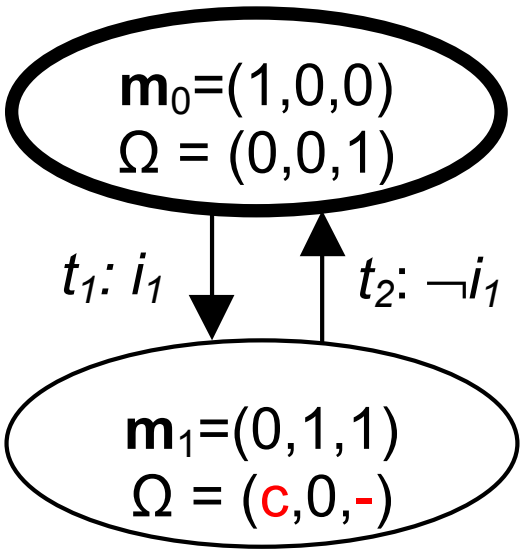
Defined and unambiguous output: There has to be a specification for the value of every output signal at every reachable marking. If two places marked at the same time assign different values to an output signal, a contradictory output setting results.

→ *Undefined and contradictory outputs can be directly read from the output functions in RG_{SIPN}.*

SIPN



RG_{SIPN}



What is Transparency?

Definition of Transparency *[Frey and Litz SMC99]*:

- At any time it must be easy and clear to see what the controller does in the moment and what it will do in the next step.
- At any time there must be the possibility to reinterpret the algorithm. This means the aim of the control algorithm must be recognizable.

Kind of criteria we seek (Lord Kelvin):

"When you can measure what you are speaking about and express it in numbers, you know something about it, but when you cannot measure it when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind."

Transparency Criteria (4 of 8)

Comments

- There should be a comment at every place and at every transition.

No trivial Input

- Defined input signals should influence the controller.

Directionality

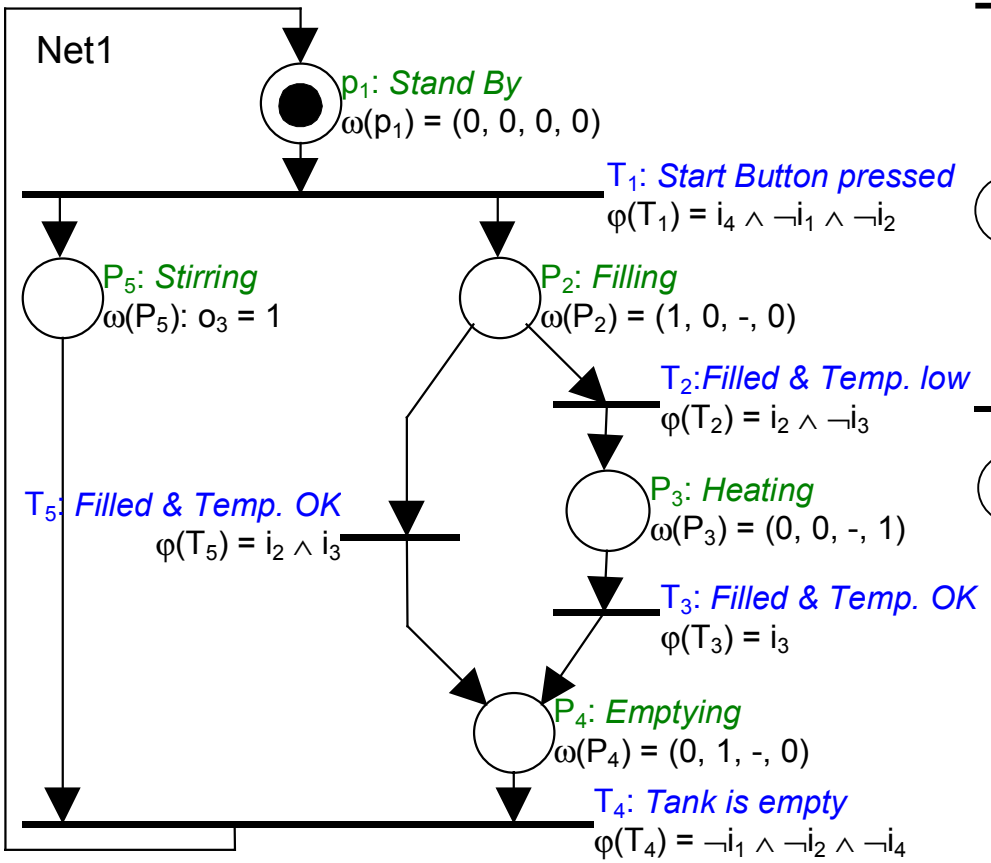
- The control flow should follow one preferred direction.

No redundant Output

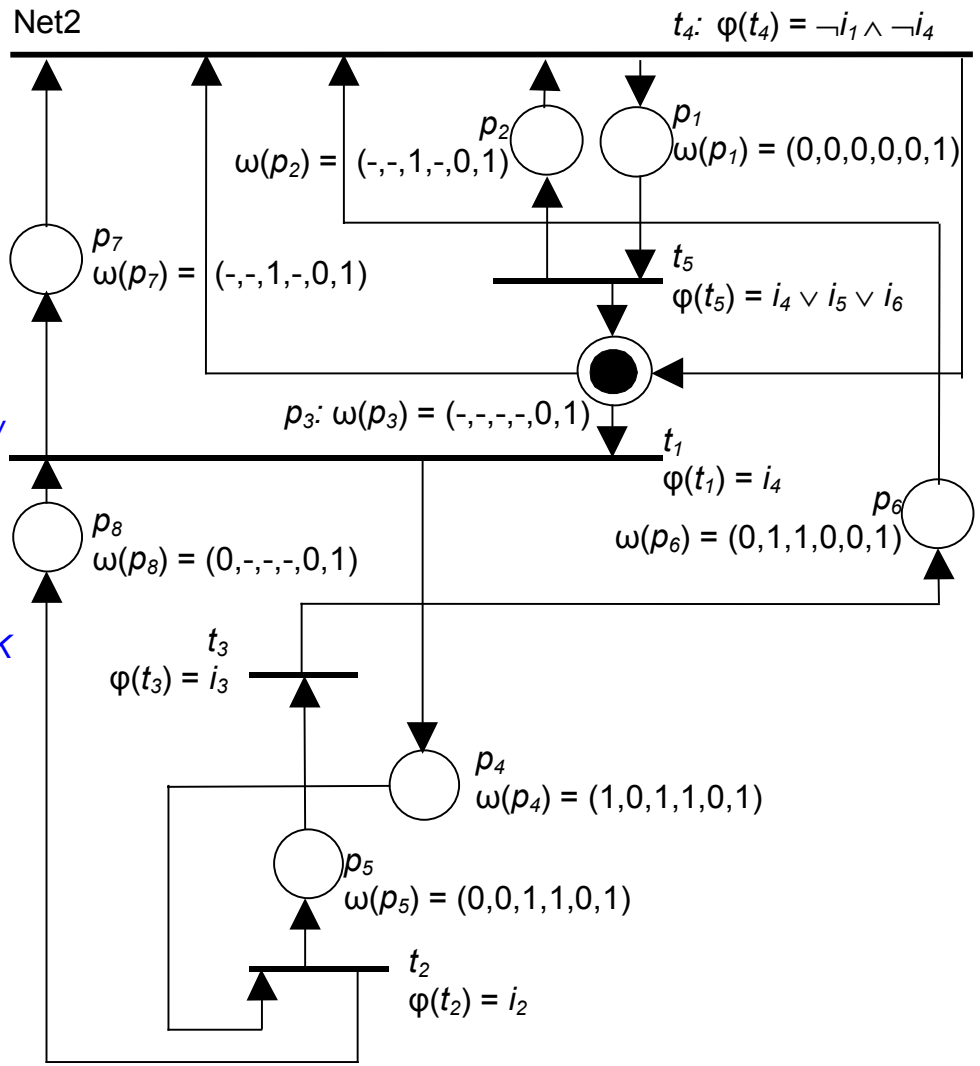
- There is redundant information if several activated places set an output signal to the same value. This hinders understanding.

TRANSPARENCY: EXAMPLE

Transparency = 1.00 (optimal)

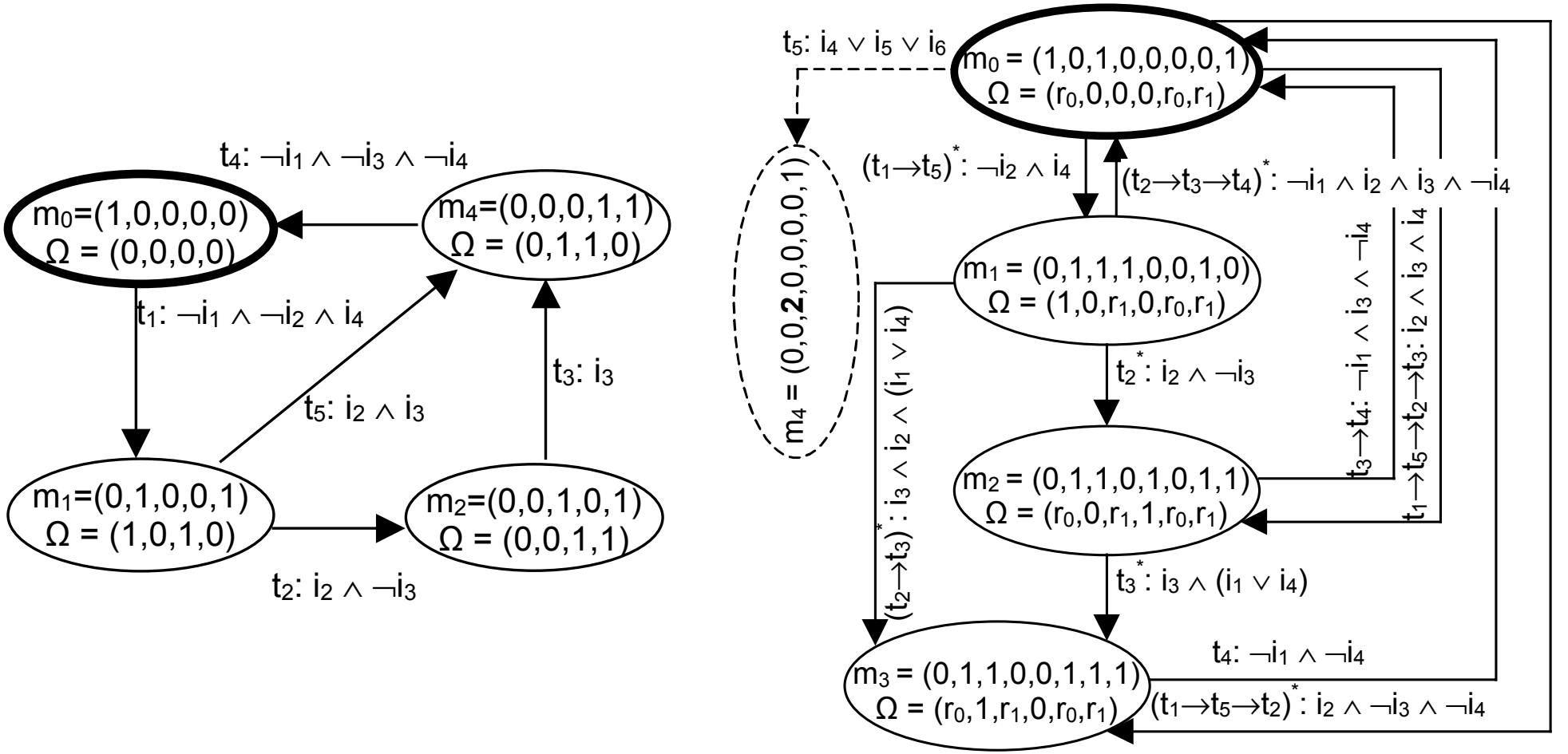


Transparency = 0.26 (worst case is 0)



TRANSPARENCY: EXAMPLE CONT.

RG $_{\text{SIPN}}$



Requirements in Code-Generation

Transparency

- Structure of SIPN has to be visible somehow in PLC code

Correctness: Especially correct Representation of Dynamics

- Concurrency
- Iteration

Efficiency

- Method of code generation
- Generated code

TRANSPARENCY OF THE IL-CODE

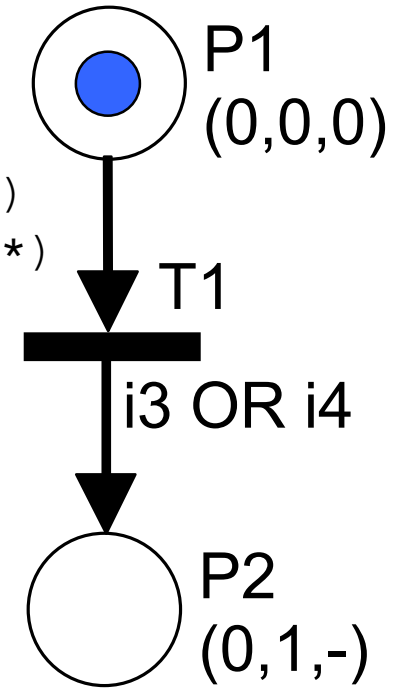
Implementation of SIPN token play

- One Boolean variable for each place

Implementation of net elements

- One-to-one correspondence between SIPN elements and code segments. Example:

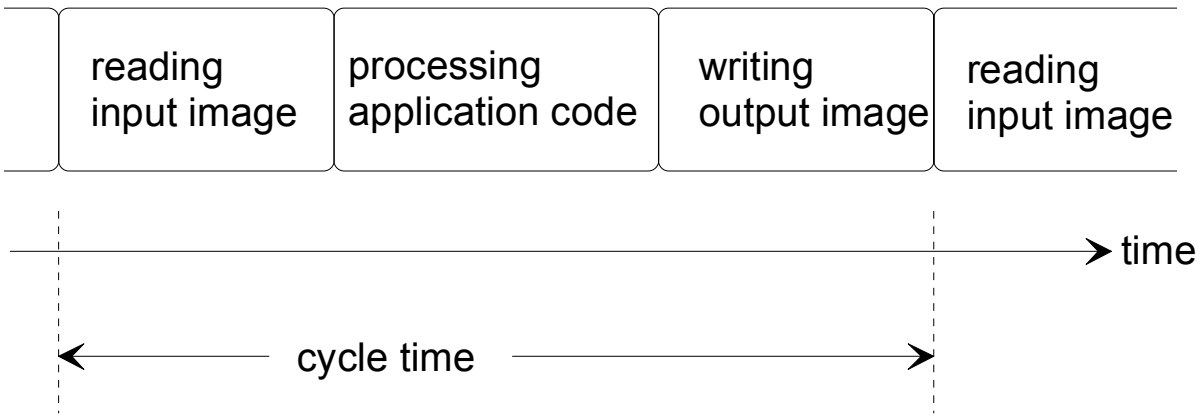
```
T1:     LD      P1      (* T1: if pre-place P1 is marked *)
        ANDN    P2      (* and post-place P2 is not marked *)
        AND(    i3      (* and firing condition is fulfilled*)
        OR      i4
        )
        R       P1      (* then unmark pre-place P1 *)
        S       P2      (* and mark post-place P2 *)
P2:     LD      P2      (* P2: if place P2 is marked *)
        S       O2      (* set O2 *)
        R       O1      (* and reset O1 *)
```



CORRECTNESS OF THE IL-CODE

Concurrency

- No problem if all transition codes are evaluated during one PLC cycle



Iteration

- Direct implementation (very inefficient code)
- Special ordering of transition codes
  - Simulative *[Jörns et al.: atp 3/1995]*
  - Analytic *[Frey: ACC 2000]*

Conclusions and Outlook

Conclusions

- SIPN allows the formal specification of logic controllers
- SIPN can be implemented on PLC automatically
- No process model needed for analysis
- Essential correctness criteria defined
- Transparency metrics defined
- All algorithms prototyped in *Mathematica*

Outlook

- Complexity metric *[Frey, Litz and Klöckner, SMC 2000 Nashville]*
- Extension to timed SIPN and to analog I/O-Signals *[Frey ADPM2000]*

Contact: frey@eit.uni-kl.de

Papers / Further Info: http://www.eit.uni-kl.de/litz/ENGLISH/frey.htm