UNIVERSITÀ DEGLI STUDI DI MILANO

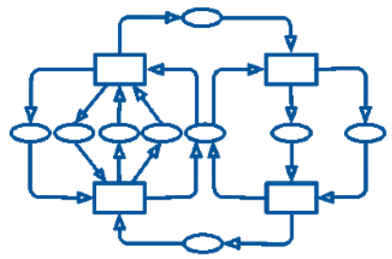
DIPARTIMENTO DI INFORMATICA

PNemu: an Extensible Modeling Library for Adaptable Distributed Systems

Matteo Camilli, Carlo Bellettini, Lorenzo Capra
{capra, bellettini, camilli}@di.unimi.it

matteo.camilli@unimi.it
http://camilli.di.unimi.it

Petri Nets 2019
Aachen, Germany June 23-28, 2019



Outline

- Introduction, Motivation, Objective
- A running example
  - self-healing manufacturing system
- PNemu modeling approach
  - Emulator + Primitives + Control-loops
- Validation & Verification
- Conclusion

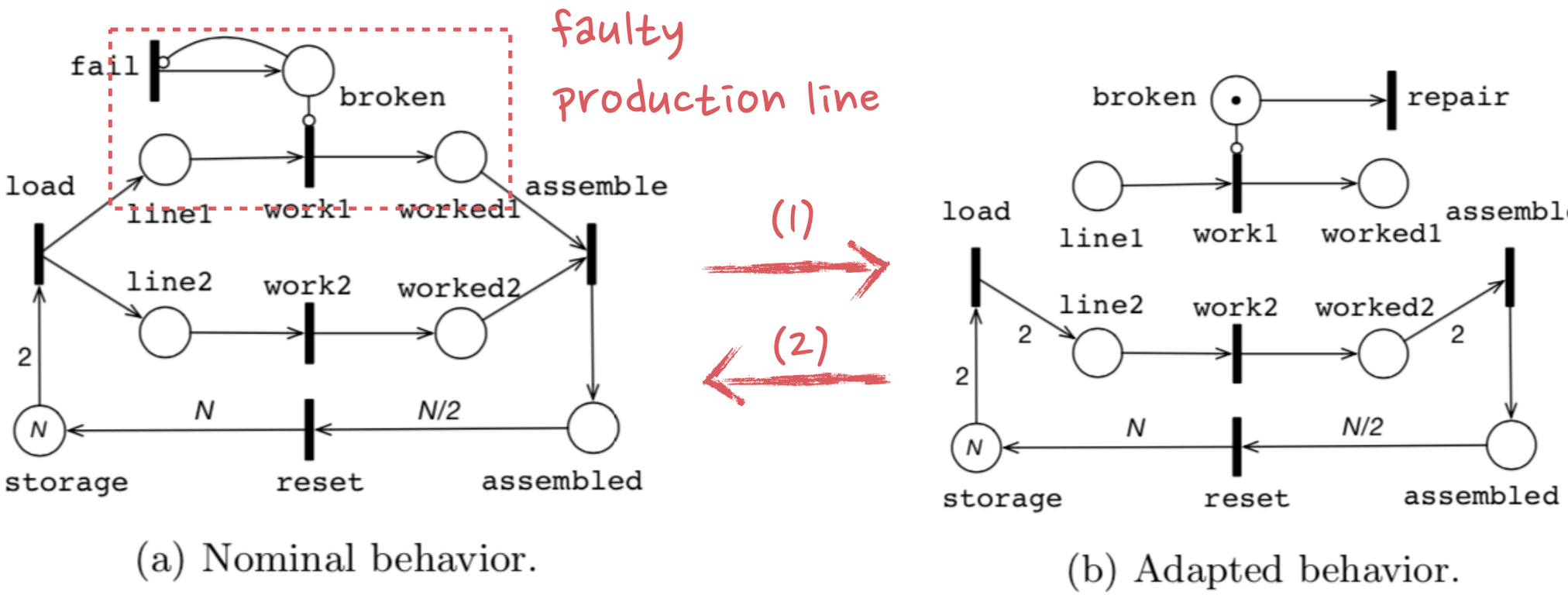
Motivations

- Distributed SA
  - Engineering distributed self-adaptive systems is hard. Formal reasoning in this context has been recognized as a major challenge in the field of SA [1]
  - Formal methods providing the ability to reason on distributed self-adaptive systems are highly demanded
- Petri Nest
  - Petri Nets represent a sound and expressive formal model for distributed discrete-event systems
  - BUT they cannot specify in a natural way structural changes that are likely to occur in adaptable and evolvable distributed applications (e.g., [2, 3])

---

1. P. Arcaini, at al. 2017. Formal Design and Verification of Self-Adaptive Systems with Decentralized Control. ACM Transaction on Autonomous and Adaptive Systems. 11, 4, Article 25 (January 2017)
2. K. Hoffmann and T. Mossakowski. Algebraic higher-order nets: Graphs and petri nets as tokens. 16th International Workshop, WADT 2002, Frauenchiemsee, Germany, September 24-27, 2002, volume 2755 of LNCS, 253–267. Springer.
3. Capra, L. and Cazzola, W. (2006). A petri-net based reflective framework for the evolution of dynamic systems. Electron. Notes Theor. Comput. Sci., 159, 41–59.

A Running Example: the self-healing manufacturing system



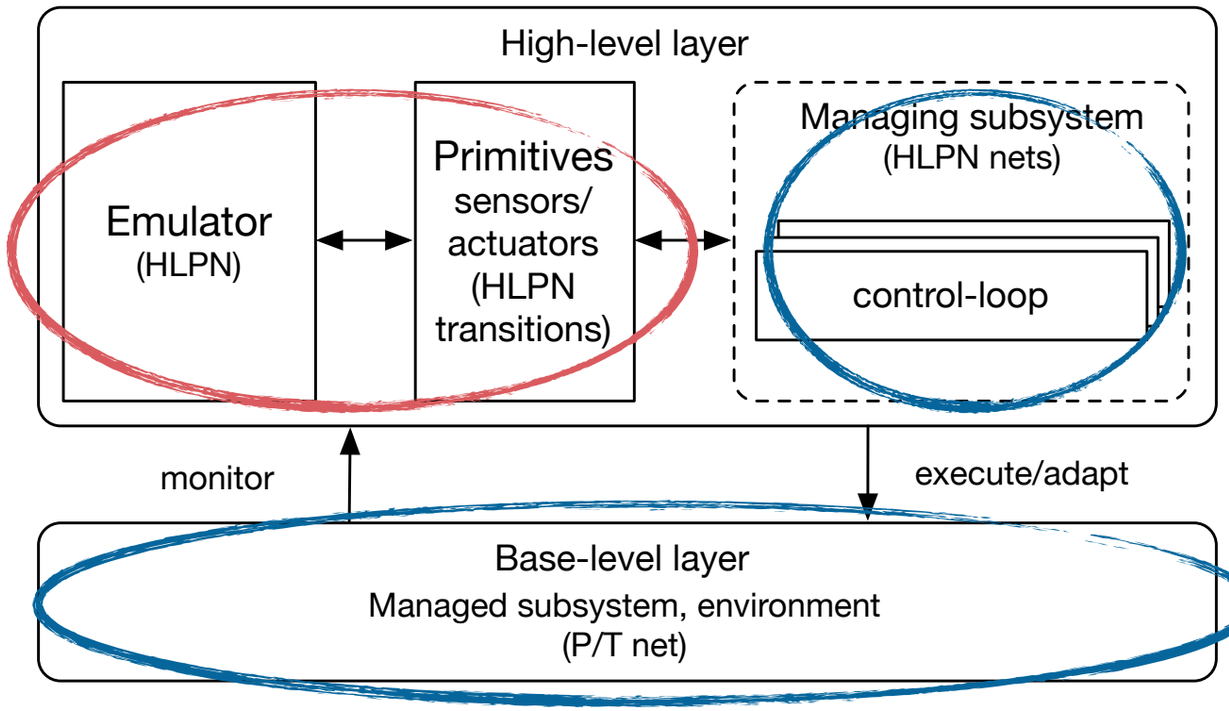
(a) Nominal behavior.

(b) Adapted behavior.

- we want to endow the MS with the ability to change its current state/structure to
  - (1) fault tolerance: detach the faulty line to keep the work going
  - (2) load balancing: distribute the load to all the available production lines
- mixing functional and adaptation aspects is HARD!
  - it often results in very complex P/T nets

PNemu: Framework Overview

fixed



user defined

- High-level layer:
  - Emulator — HLPN [4,5] model able to execute a P/T net
  - Primitives — HLPN transitions that read/change the encoded P/T net model (sensors + actuators)
  - Control-loops (managing subsystem) — HLPNs defining the adaptation aspects

4. Systems and software engineering - High-level Petri nets - Part 1: Concepts, definitions and graphical notation. Standard, International Organization for Standardization, Geneva, CH (Dec 2004), ISO/IEC 15909-1:2004

5. Pommereau, F.: Snakes: A flexible high-level petri nets library (tool paper). In: Devillers, R., Valmari, A. (eds.) Application and Theory of Petri Nets and Concurrency. pp. 254–265. Springer International Publishing, Cham (2015)

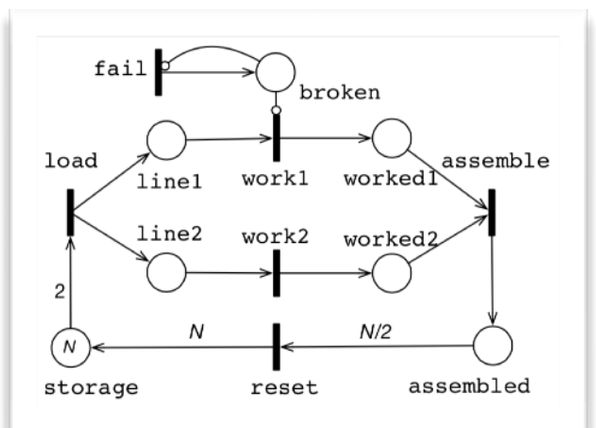
Emulator instance

PNemu code snippet

```
1. from pnemu import PT, Emulator
2.
3. pt = PT('ms', 'ms-nominal.pnml')
4. emulator = Emulator( pt )
5.
6. emulator.get_marking().get( 'mark' )
7. # MultiSet( ['storage'] * 10 )
```

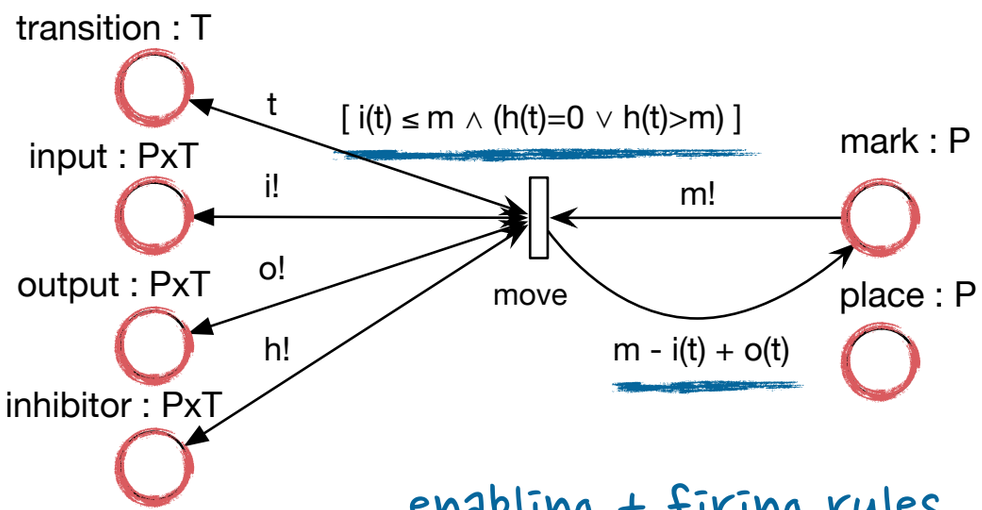
(3)

base-level layer creation



(4)

Emulator instance



enabling + firing rules

Primitives

- HLPN **transitions** connected to specific structural elements of the Emulator
  - read/modify the state/structure of the base-level layer
  - PNemu provides a set of pre-defined **read (sensors)** / **write (actuators)** primitives

- Examples:

Sensor

PNemu code snippet

```
1. signature = "lib.getTokens(p_) -> m(p_)"
2. entry = LibEntry(
3.     signature,
4.     [ Place('M') ],
5.     [ ('M', signature, Flush('m')) ],
6.     [ ('M', signature, Flush('m')) ])
7. READ_LIB.update({function_name(signature) : entry})
```

Actuator

PNemu code snippet

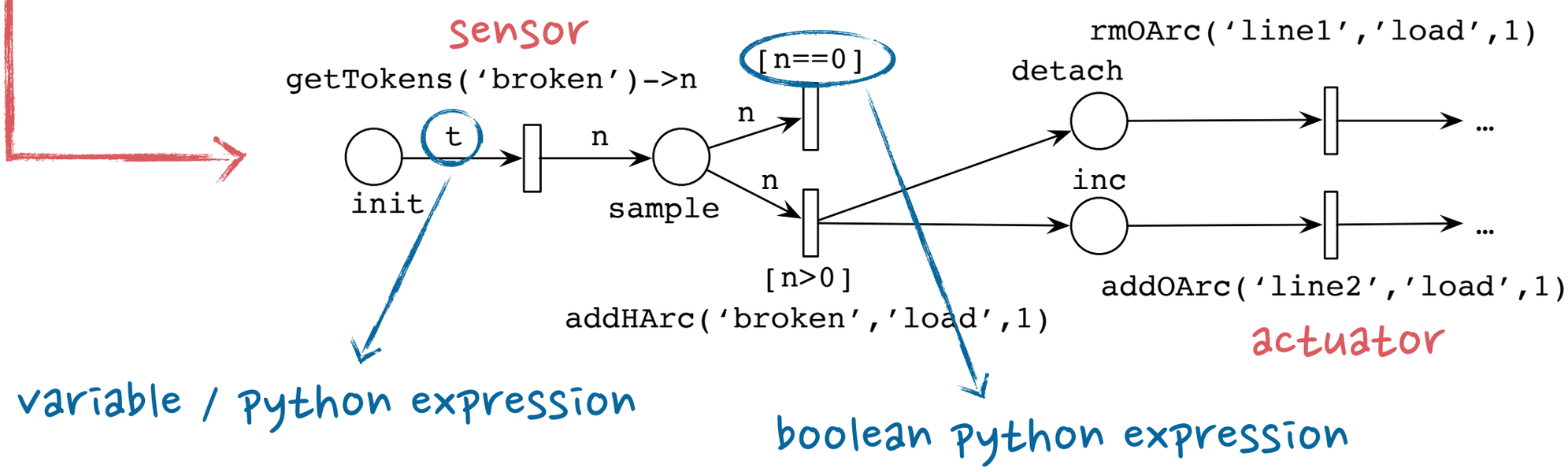
```
1. signature = "lib.addPlace(p_)"
2. entry = LibEntry(
3.     signature,
4.     [ Place('P') ],
5.     [ ],
6.     [ ('P', signature, Expression('p_')) ])
7. WRITE_LIB.update({function_name(signature) : entry})
```

Managing Subsystem

- user-defined HLPNs (one for each **adaptation concern**)

PNemu code snippet

```
1. loop = FeedbackLoop(name='fault-tolerance')
2. loop.add_place(name='init')
3. loop.add_place(name='sample')
4. read_primitive = 'lib.getTokens("broken")->n'
5. loop.add_transition(name=read_primitive)
6. loop.add_input_arc(src='init', dst=read_primitive, Variable('b'))
7. loop.add_output_arc(src=read_primitive, dst='sample', Variable('n'))
8. …
```



Validation and verification activities

PNemu code snippet

```
1. net = AdaptiveNetBuilder(emulator)                                    \
2.     .add_loop(control_loop=loop, initial_places=['init'])    \
3.     .add_loop(control_loop=loop2, initial_places=['init2']) \
4.     .build()
```

- Validation
  - token game by using the PNemu API; step-by-step check:
    - marking/structure of both the managed and the managing systems
    - enabled transitions of both the managed and the managing systems
    - active control-loops
- Formal Verification
  - model checking by using external tools, e.g. the Neco LTL model checker [6]:
    - deadlock freedom $\Box(\neg \texttt{deadlock})$
    - safety + liveness properties + invariants

      $\Box(\texttt{marking}(\texttt{'M'}) >= [\texttt{'broken'}] \rightarrow \neg\Box(\neg\texttt{marking}(\texttt{'M'}) >= [\texttt{'broken'}]))$
    - adaptation integrity constraints

      $\Box(\texttt{card}(\texttt{marking}(\texttt{'init'})) > 0 \rightarrow \Diamond(\texttt{card}(\texttt{marking}(\texttt{'detached'})) > 0))$

6. Fronc, L., Pommereau, F.: Building Petri nets tools around Neco compiler. In: Proceedings of PNSE'13. pp. 1–7 (06 2013)

Conclusion

- We are trying to support design and validation/verification of distributed self-adaptive systems with decentralized adaptation control (recognized as a major challenge in SA)
- We leverage HLPNs applying clear separation of concerns to model SA
- Our approach gets solid foundations from well-established formal methods (HLPNs) and leverages existing analysis techniques and tools (such as SNAKES and Neco)

- PNemu is available at: https://github.com/SELab-unimi/sn-based-emulator 

- Future work
  - DSL to describe the managing subsystem (using directly HLPNs may be tricky)
  - enhance the verification activity providing the ability to check that multiple control loops do not lead to conflicting adaptation decisions

UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI INFORMATICA

Thank you

Questions?

Matteo Camilli, Carlo Bellettini, Lorenzo Capra
{capra, bellettini, camilli}@di.unimi.it

Speaker: http://camilli.di.unimi.it