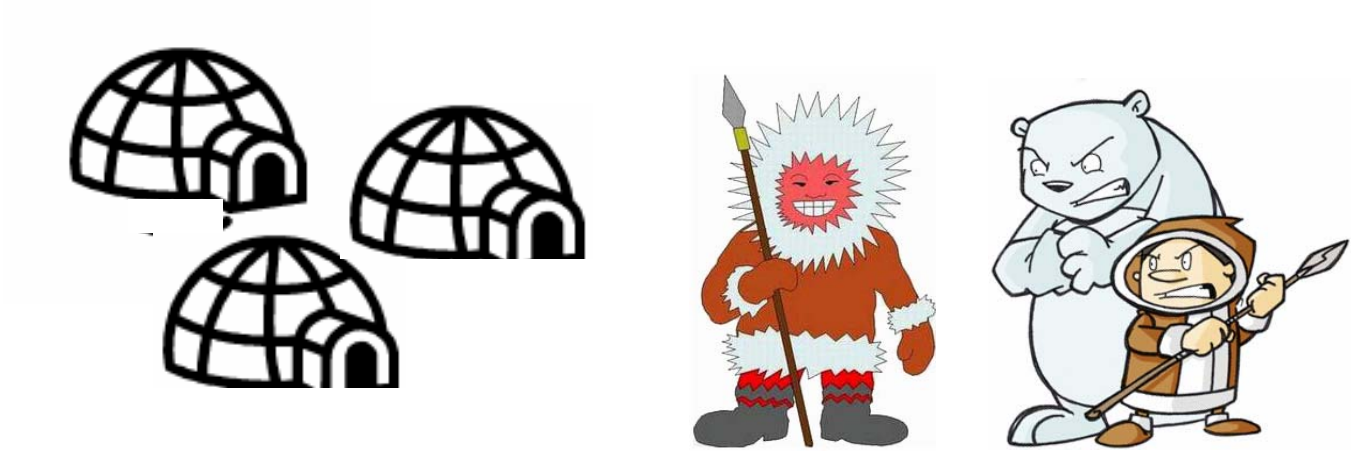
An illustrative series of HLPNs:

– ***Dijkstra's Mutex examples***

– ***in Ben-Ari's* 1982 *staging***



Two families lived near the North Pole.
Sometimes one of the hunters of a family would **accidentally kill** a hunter of the other family – due to bad visibility (snow storms, hail and fog). ☹

Mutual exclusion

Common two-family conference

→ **Mutual exclusion: never hunt simultaneously**

For both families, hunting is a **critical section**.

Each party stays

in its **critical section**

only in a **limited time interval.**

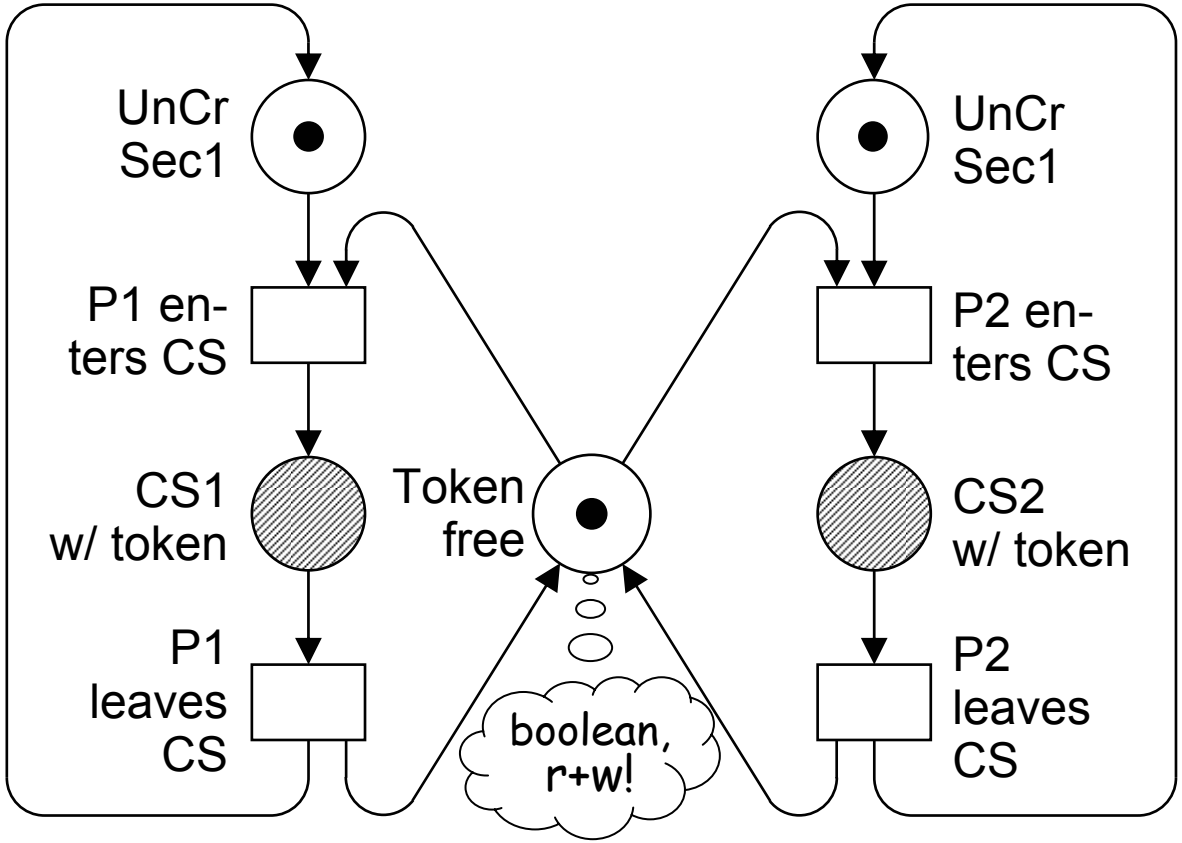
Mutual exclusion is ensured by means of

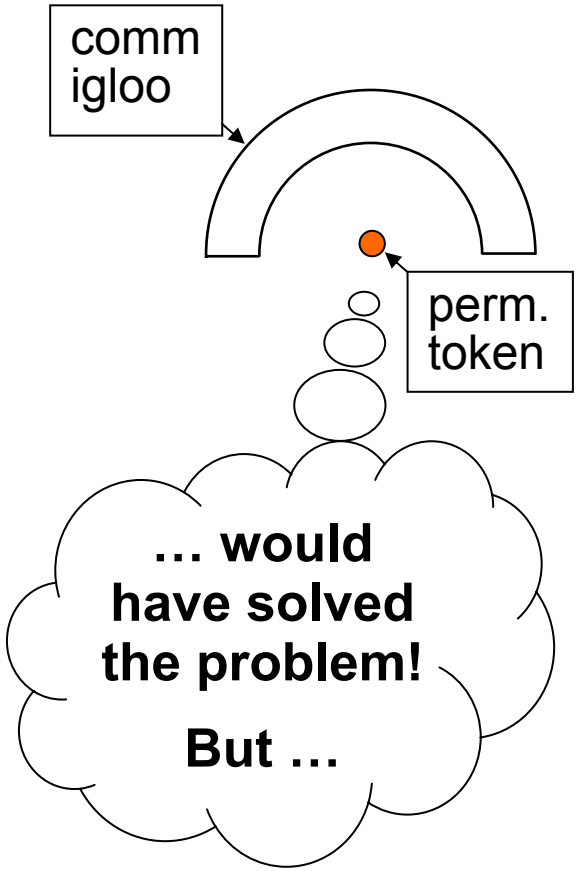
one or more **communication igloos.**

MutEx0

atomic (**reading&writing**)

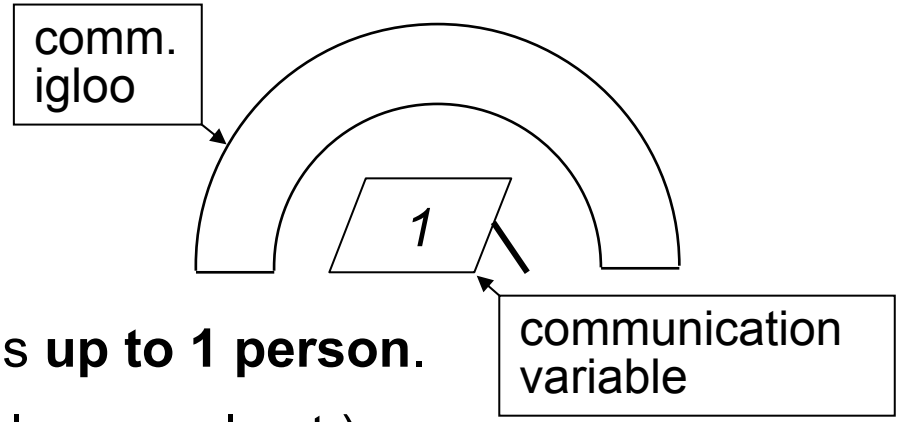
– the “permission token” approach





Additional problem!

These poor people
could only think up
atomic reading and **atomic writing**:



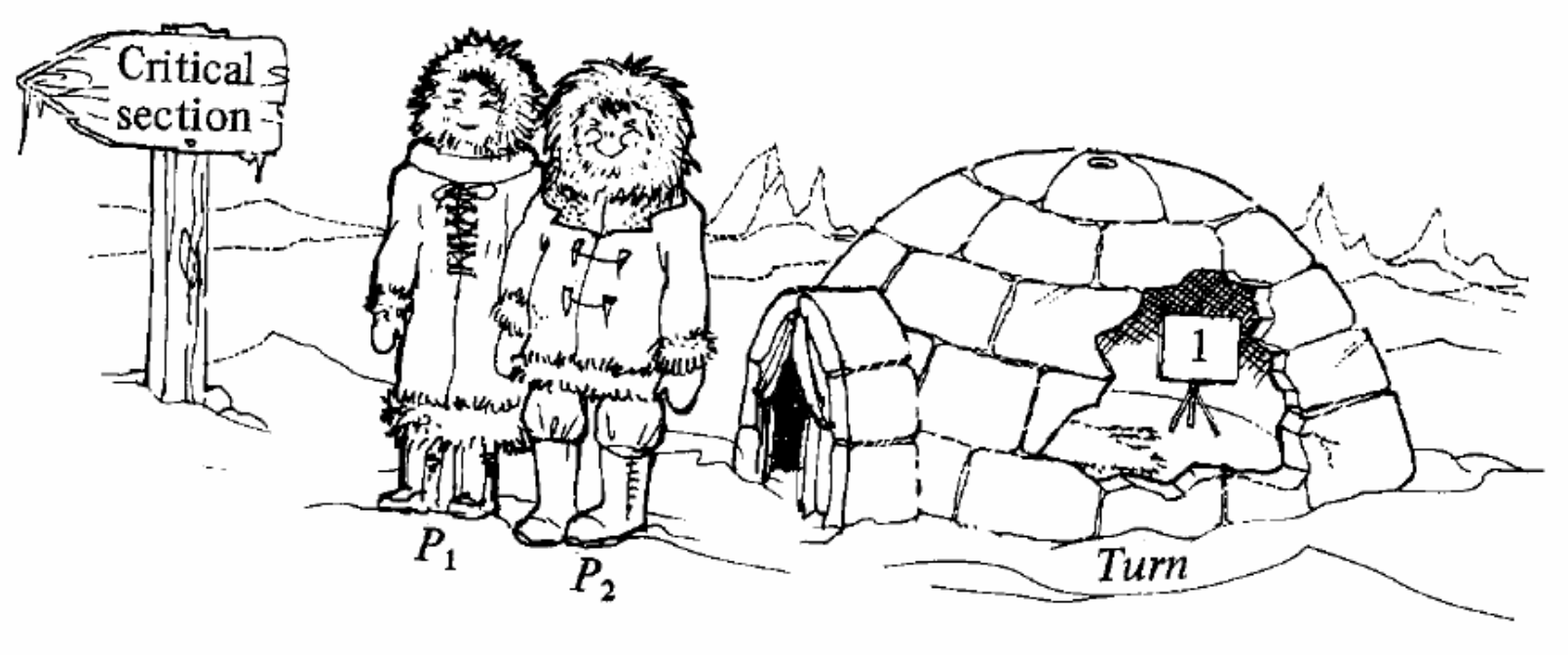
Communication igloo holds **up to 1 person**.

Uses: (Crawl in, read value, crawl out)

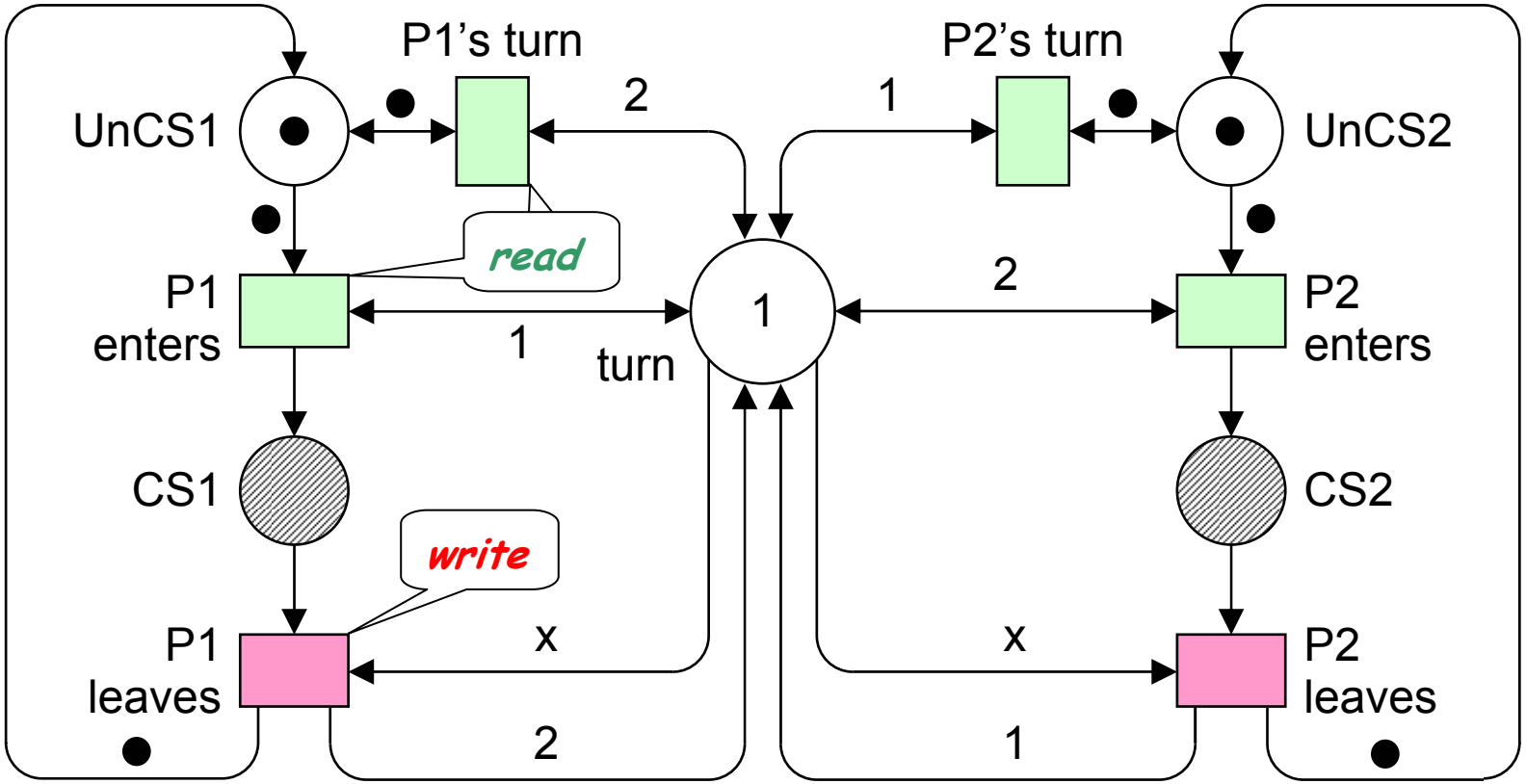
XOR (Crawl in, (over)write value, crawl out)

Could the accidents be stopped?

MutEx1: 1 igloo telling “my turn – your turn”



MutEx1: 1 igloo telling "my turn – your turn"



Tragically, P1 were invited by their uncle who had caught a whale.
When they returned after 3 months, some of P2 had starved.☹

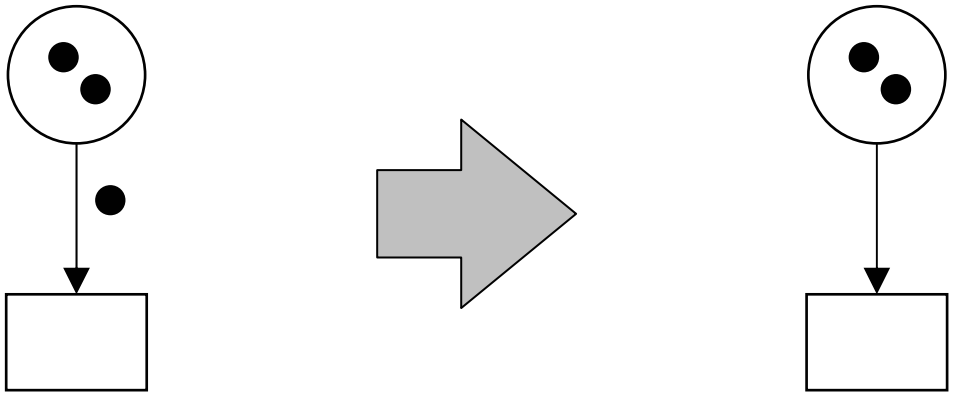
starvation!

Tiny simplification

We omit constant one-token expressions ● at arcs –

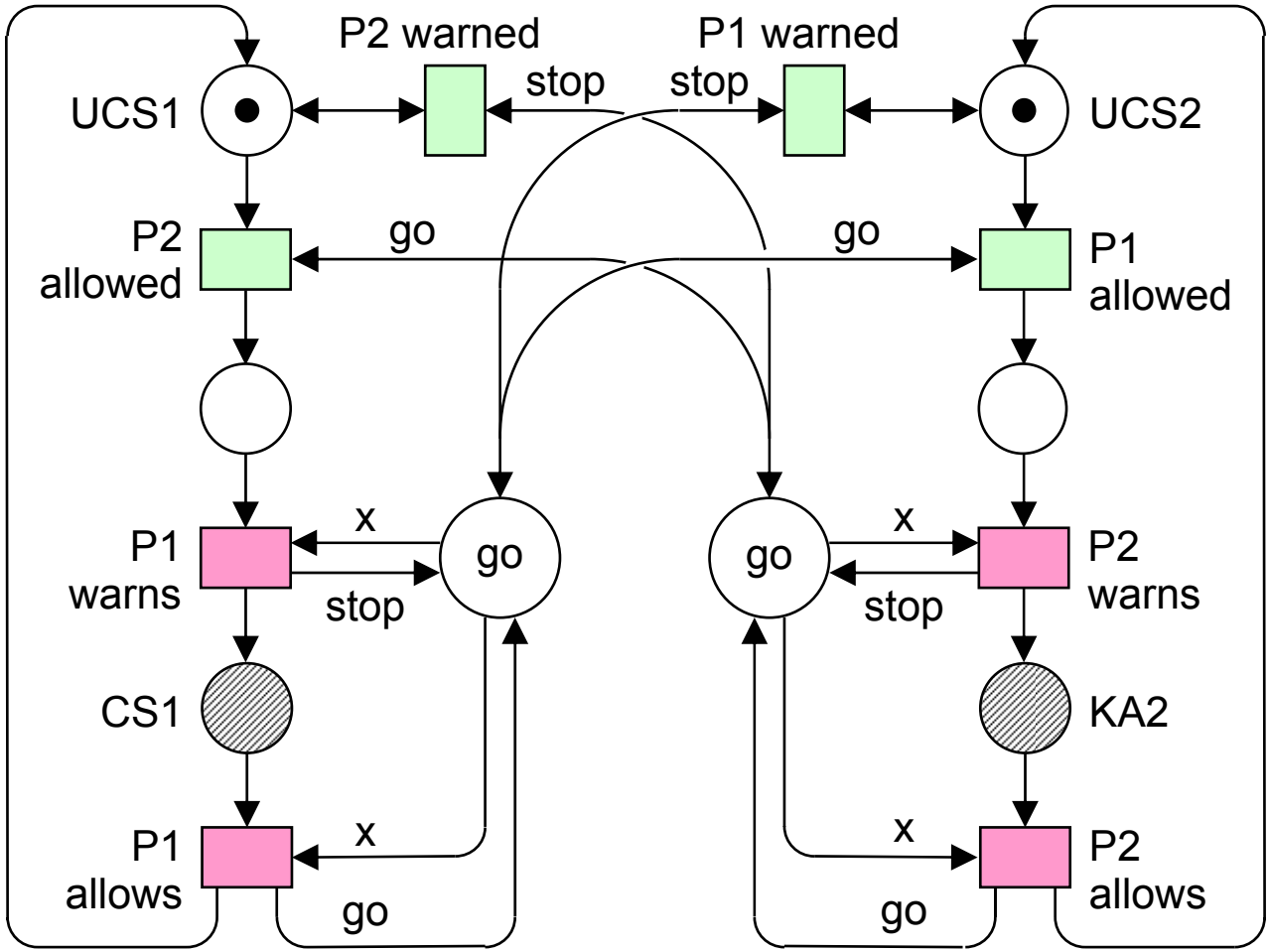
this becomes the **default arc inscription**.

Cf. the “1” in PT-systems.



MutEx2: 2 “warning” igloos: check there – then warn here

MutEx2: 2 “warning” igloos: check there – then warn here

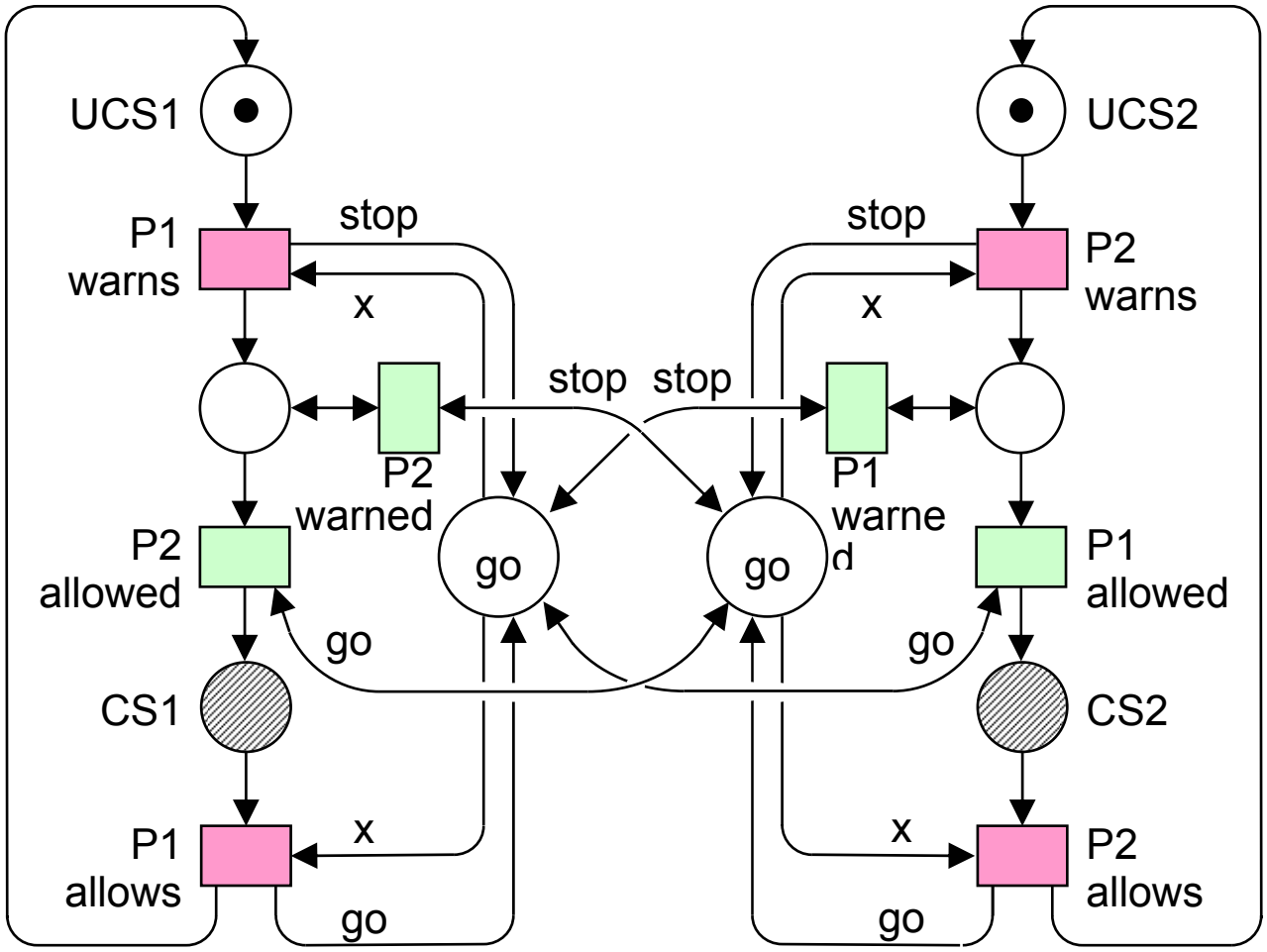


Tragically, one day
P1 and P2
started acting
concurrently,
at about
the same speed –
and a deadly
accident
occurred!

☹

unsafe!

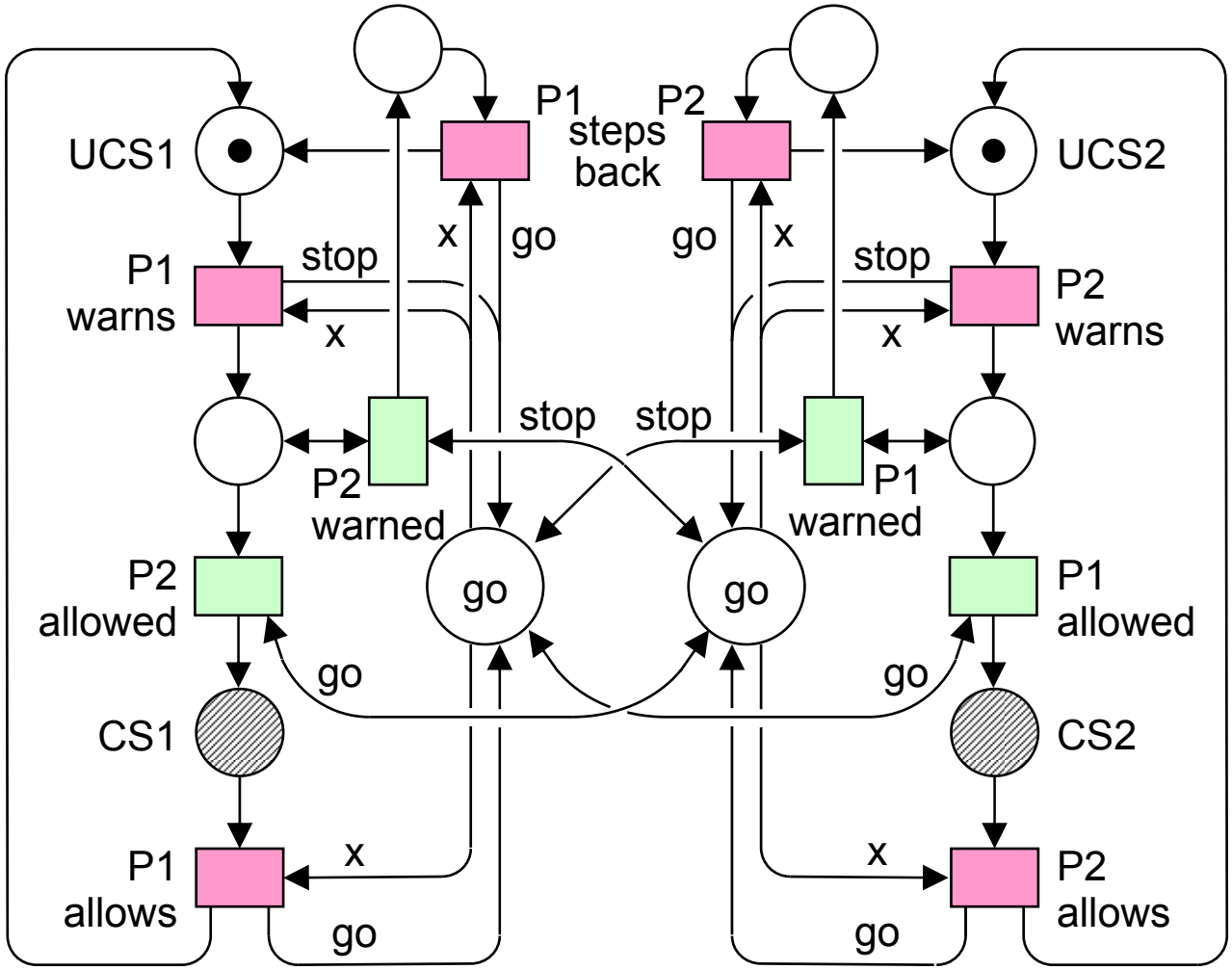
MutEx3: 2 “warning” igloos: warn first – then check there



Tragically, one day
P1 and P2
started acting
concurrently,
at about
the same speed –
and many starved!
☹

lifelock!

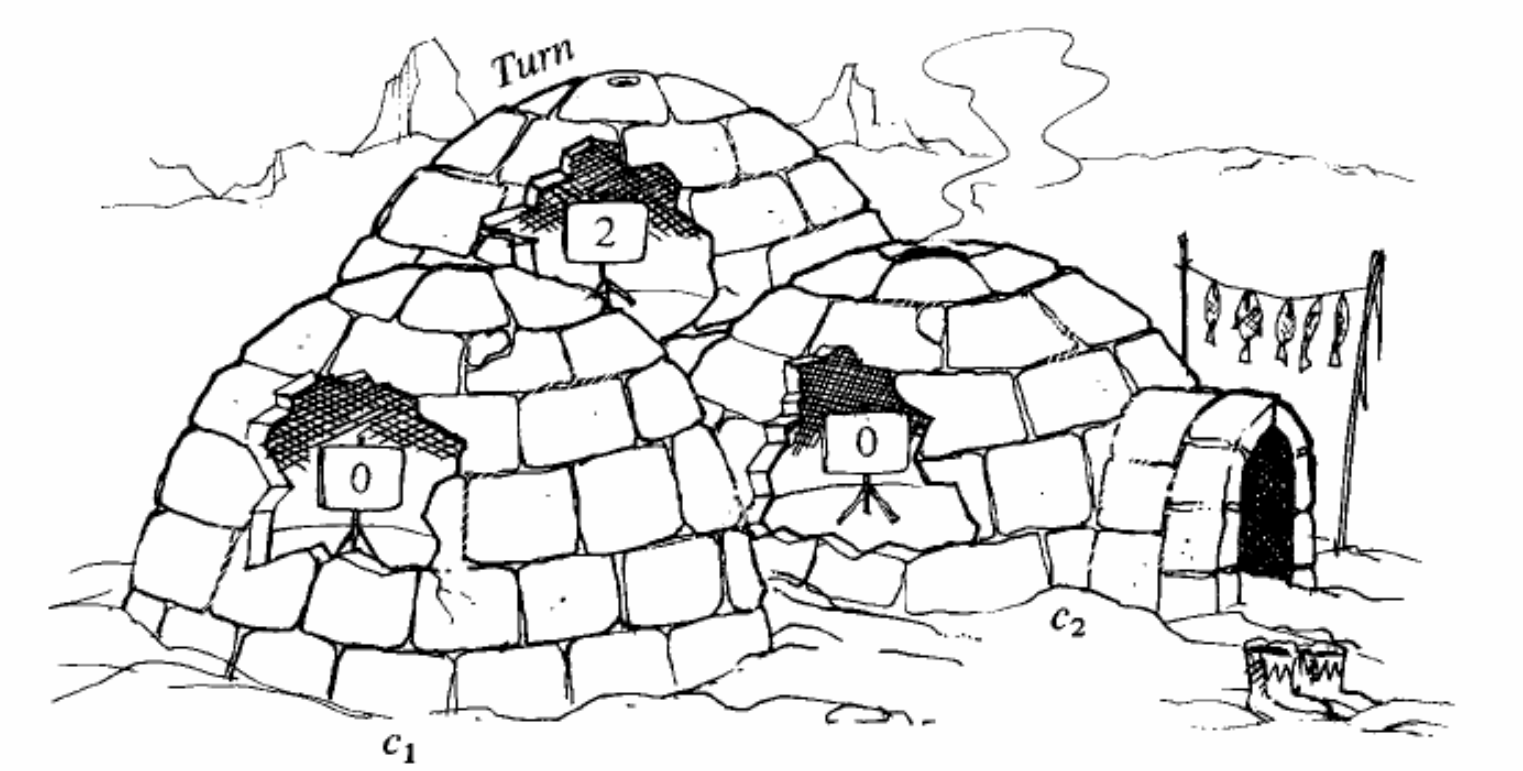
MutEx4: MutEx3 & if warned, step back temporarily



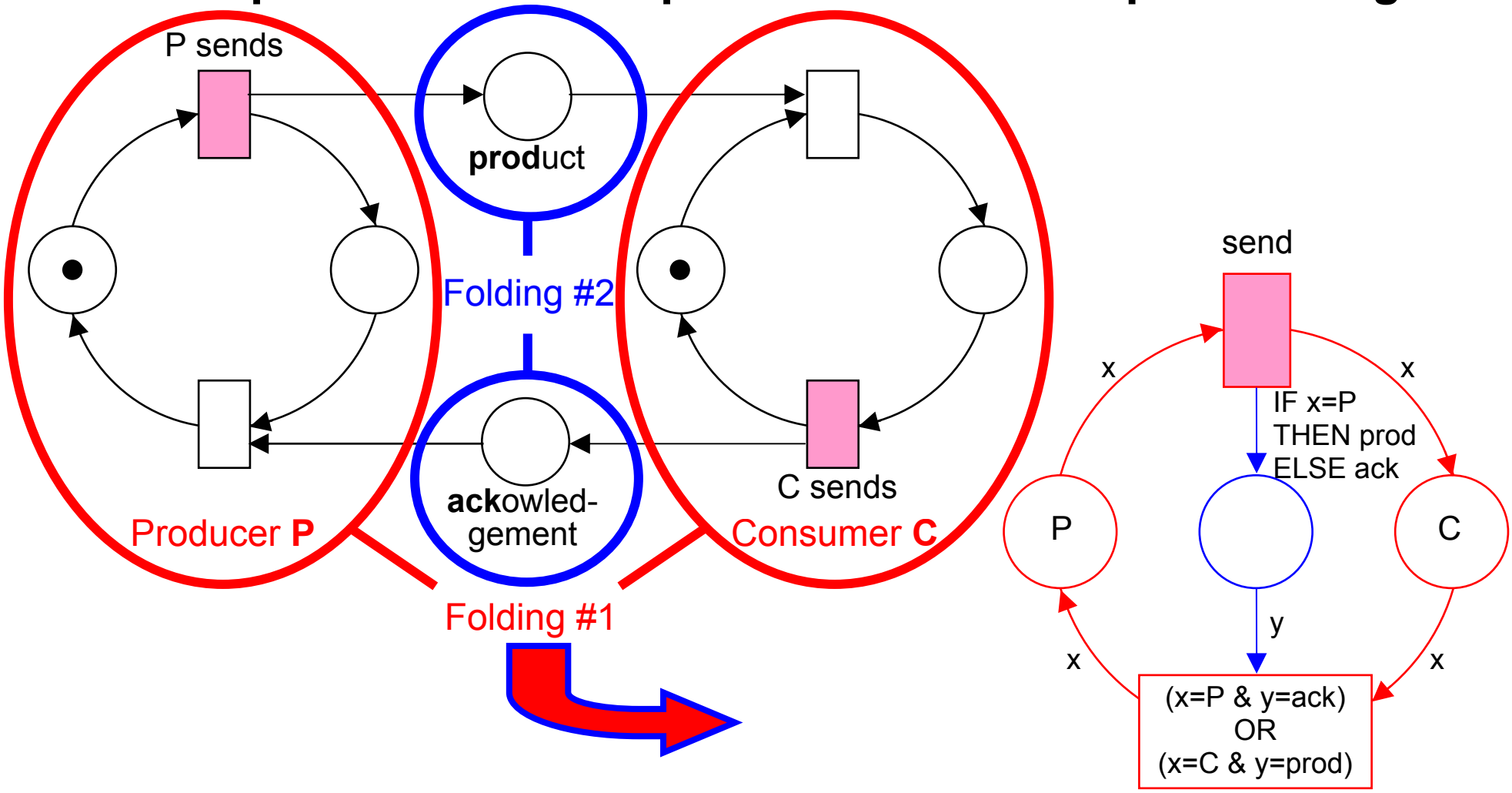
Tragically, one day
P1 and P2
started acting
concurrently,
at about
the same speed –
and many starved!
☹

lifelock!

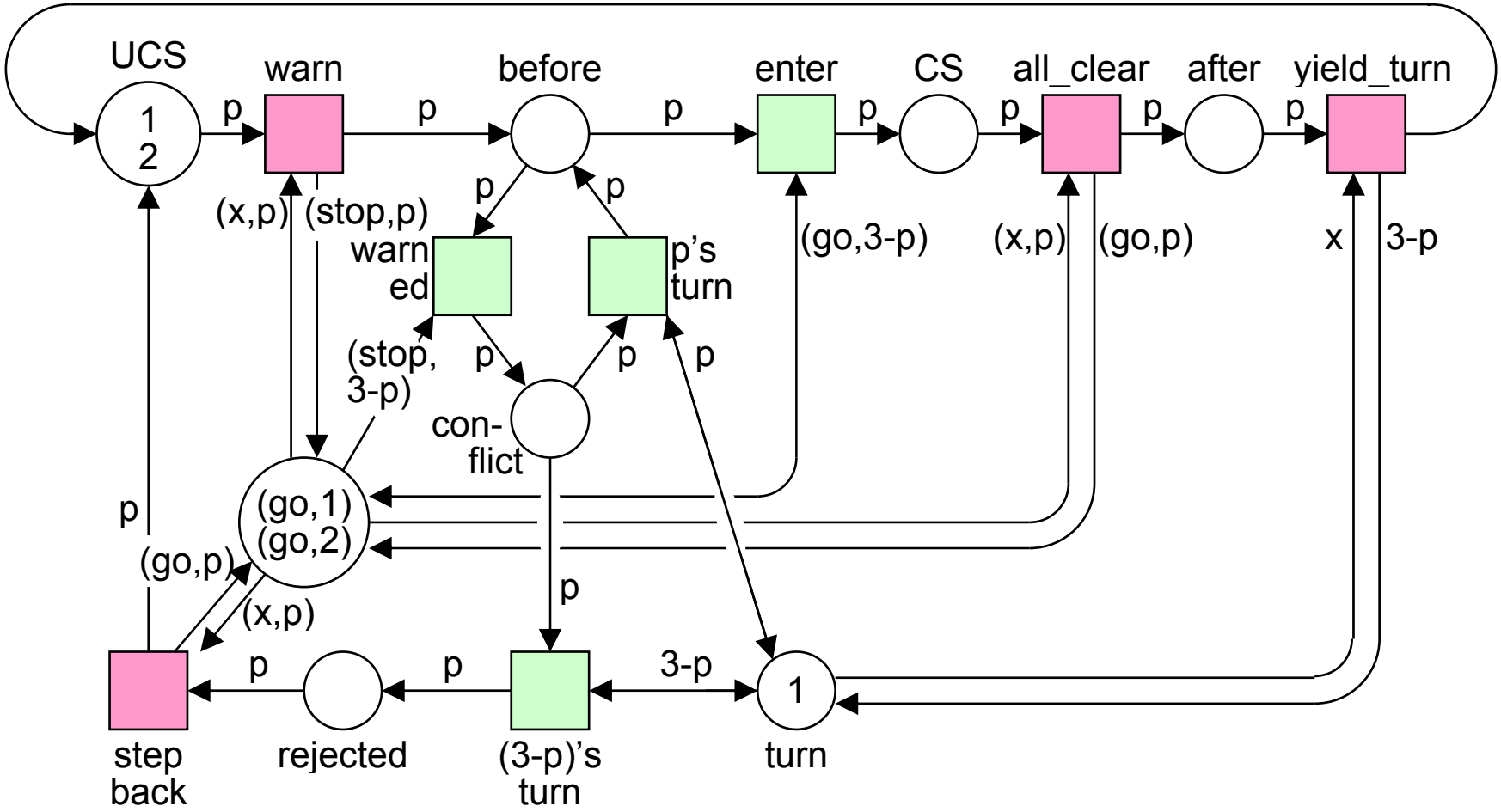
MutEx5: 2 “warning” + 1 “turn” igloos



HLPN representation compactification technique: *Folding*



MutEx5: 2 “warning” + 1 “turn” igloos



… and they never had another hunting accident again!

Dijkstra: "Dekker's algorithm"

Reachability Analysis
→ (about 36 states) → correct!

Analysis methods for HLPNs

- **Reachability analysis** works in total analogy to PN systems is, and satisfactorily so if finite or even small.
- An analogon to coverability analysis to **decide boundedness** is in general impossible. It could be used to solve the halting problem: emulate the possible "states of a program execution" and count the statement executions in an HLPN – which has been proven impossible.
  It may work in very special cases, though, with properly adapted definitions.

Linear analysis still works, though with more complicated notions and proofs (→ K. Lautenbach, A. Pagnoni: On the Various High-level Petri Nets and Their Invariants, *EATCS-Bulletin*, February 1984).

- As shown in the packet-letter-example, cleverly chosen firing **invariants**, possibly even if independent from linear analysis, may be obtained with general mathematics applied creatively, and may help to prove system properties.

HLPN Literature

B. Baumgarten: cf. Petri Net Basics

W. Brauer, W. Reisig, G.Rozenberg, (Eds): *Petri Nets : Central Models and Their Properties*, Lecture Notes in Computer Science ,Vol. 254, Springer, 1987.

W. Brauer, W. Reisig, G.Rozenberg, (Eds): *Petri Nets : Applications and Relationships to Other Models of Concurrency*, Lecture Notes in Computer Science ,Vol. 255, Springer, 1987

K. Jensen: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. 2nd corrected printing. Volumes 1+2, Monographs in Theoretical Computer Science, Springer, 1997

W. Reisig: Petri nets and abstract data types, *Theoretical Computer Science* 80, pp 1-34, 1991

M. Ben-Ari: *Principles of Concurrent Programming*, Prentice-Hall, 1982. The illustrating story was dropped in his 1990's *Principles of Concurrent and Distributed Programming* (I wonder why – political correctness? too frivolous?)