

PhD Summer School
Formal Methods for System Analysis in Informatics
Druskininkai, LT, May 2007

Petri Nets
with Timing Notions

Bernd Baumgarten

Fraunhofer SIT
Rheinstr. 75, 64295 Darmstadt, Germany
bernd.baumgarten@sit.fraunhofer.de
<http://private.sit.fhg.de/~baumgart/>
+49 6151 869263

Purpose and realization of timing

Ultimate purpose:

system specification → **timed observation sequences**

*Technical approach – **Timer nets:***

Specify **temporal conditions**
for the **occurences** of **transitions**
in **place-transition systems**

It could be done with **HLPNs**
instead, but let's stay **simple!**

Timing relationships

Set **upper** and **lower time limits**

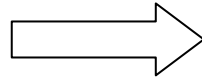
for the **occurrence** of a transition

with respect to

- prior occurrences of **other transitions**,
- prior occurrences of **the same transition**,
- **absolute** time (in the physical world),
- time **differences** of the above.

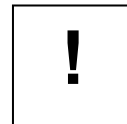
Transition types

Analysis of examples



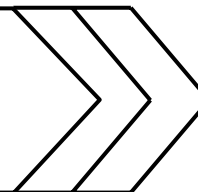
3 kinds of transitions needed

- MAY-transitions
- MUST-transitions
- ASAP-transitions



Occurrences do not consume time!
Time passes chiefly (?!) between firings.

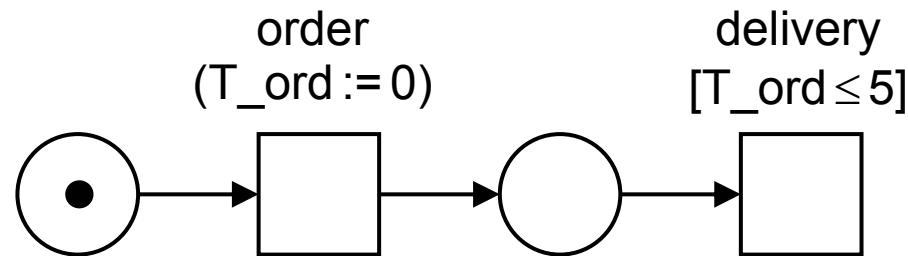
We illustrate the transition types and timing relationships in parallel by means of intuitively understandable examples.



Timer net example: Delivery Deadline

... introduces

- timing relative to earlier occurrence moments of *other* transitions
- *MAY* transitions



Delivery deadline

Informally: *order* **may** happen once, at any time.
delivery **may only** occur ≤ 5 time units after *order*, **else not at all**:
 “Deliver in time, or forget it.”

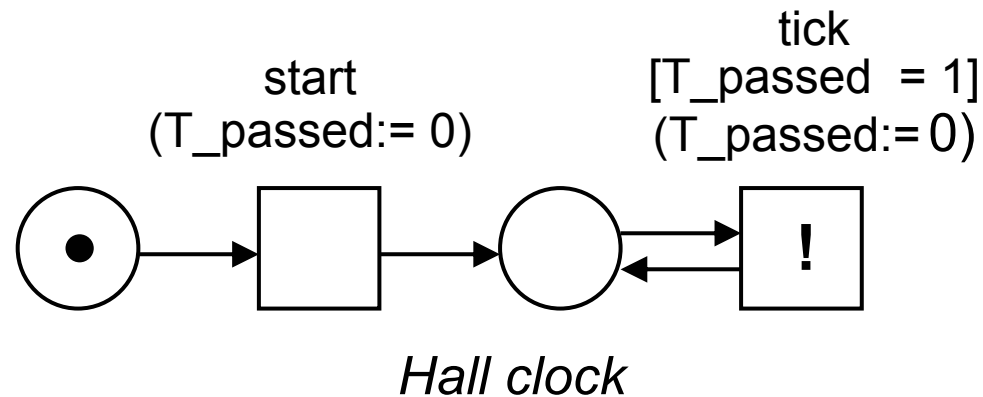
Notation: T_ord = time shown by timer (clock) T_ord , a real number.
 $(T_ord := 0)$ = a clock-setting (timer start).
 $[]$ encloses timing conditions.

Similarly: opening hours, legal deadlines, ...

Timer net example: Hall Clock

... introduces

- timing relative to the last occurrence moment of the *same* transition
- our first *MUST* transition



Informally: It ticks every time unit, initially 1 time unit after start.

Notation: ! *tick* is a **mandatory**, a *MUST* transition; it is guaranteed to fire within its time limits (unless disabled at the end).

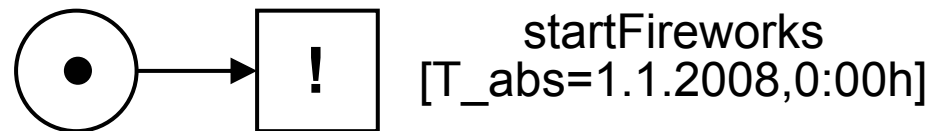
Reminder:

Promises without time limits are void.

Timer net example: Fireworks

... introduces

- timing relative to real world time



New year's fireworks

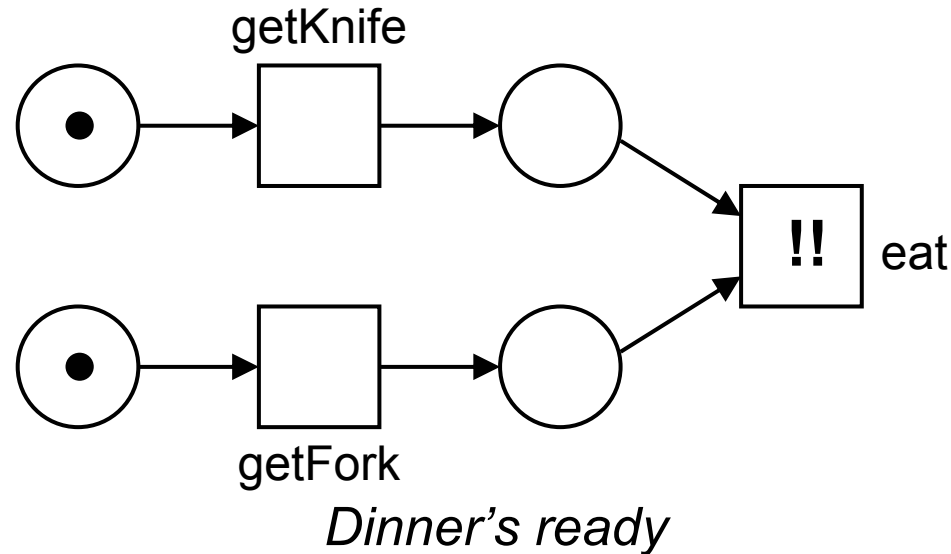
Informally: A big firework shall start at midnight, 1.1. 2008.

Notation: T_{abs} = real world time, e.g. as GMT.

Timer net example: Dinner's ready

... introduces

- our first ASAP transition (As Soon As Possible)

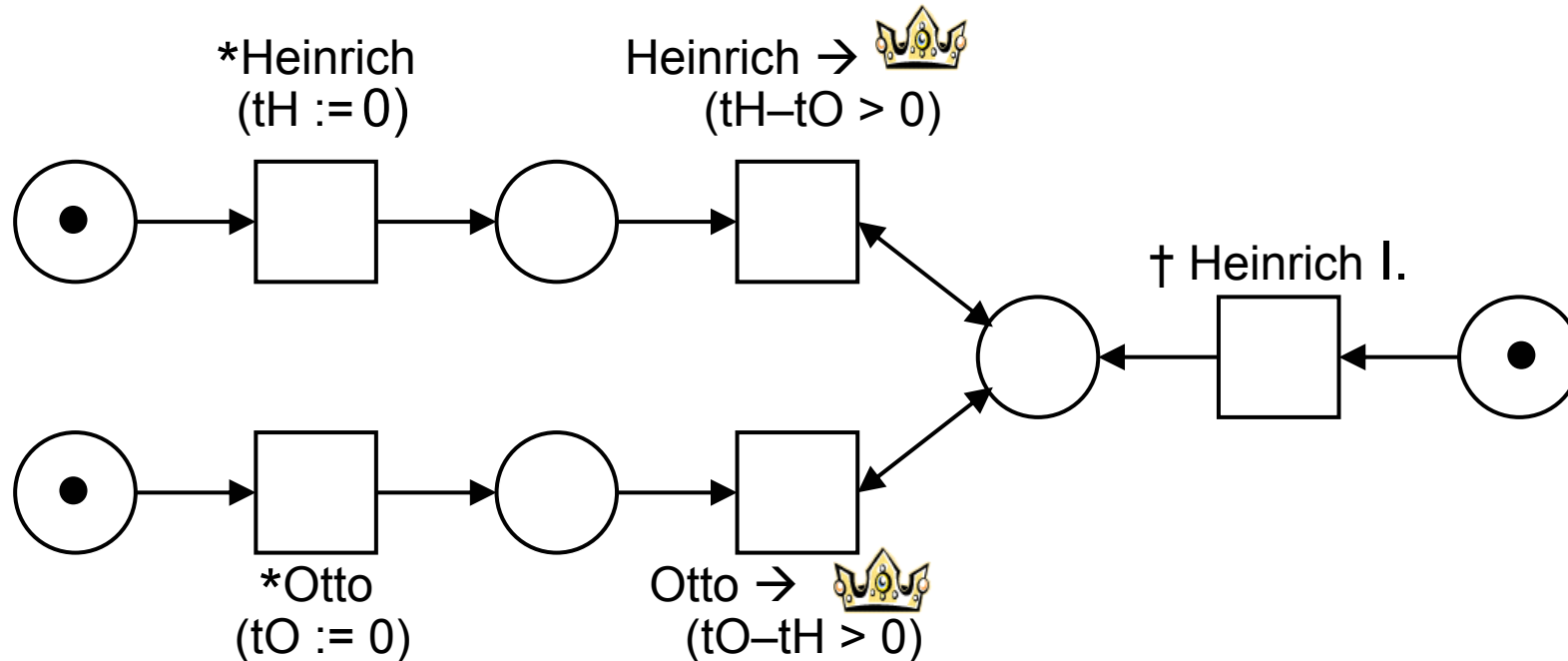


Informally: Eat as soon as in possession in knife and fork.

Notation: **!!** The transition is **immediate** or **ASAP**:
eat must fire as soon as enabled w.r.t. **tokens and time**
 (unless disabled right then, or time condition not true).

Timer net example: Primogeniture

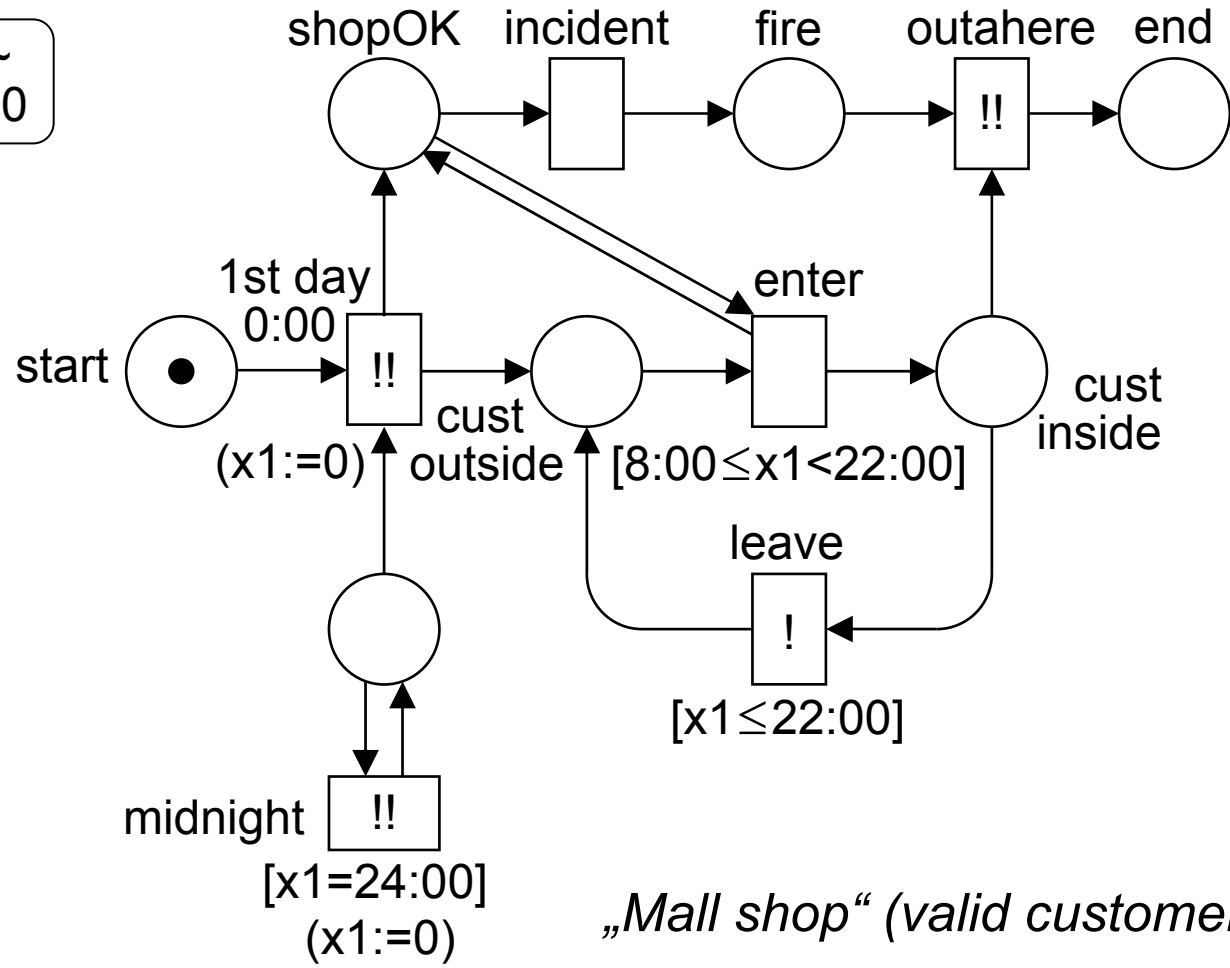
... introduces • the use of timer differences



Informal description: The first-born son of King Heinrich I (“the Bird-Catcher”) was the first candidate upon the throne after the King’s death. Thus Prince Otto became King Otto and later Emperor Otto the Great, while Prince Heinrich did not become king, quarrelled with Otto and was later named Heinrich “the Quarrelsome”.

MAY + MUST + ASAP – a comprehensive example

world time $x_0=0 \sim$
opening day, 0:00



„Mall shop“ (valid customer behaviour)

Time value range \mathbb{T}

Instead of \mathbb{R} , it might be desirable to use e.g.

- a **dense** unbounded set \mathbb{T} of non-negative numbers, including 0, where timer **differences** may, of course, be **negative**.

Thus, between any two numbers $a < b$ in the set there exists another one, $a < y < b$ – and hence infinitely many.

Theoreticians prefer the **reals** $\mathbb{R}_{\geq 0}$,

or the **rationals** $\mathbb{Q}_{\geq 0}$,

while the practically most widely used range is $\mathbb{D}_{\geq 0}$, the natural multiples of the integer powers of 10, i.e. the **terminating decimals**.

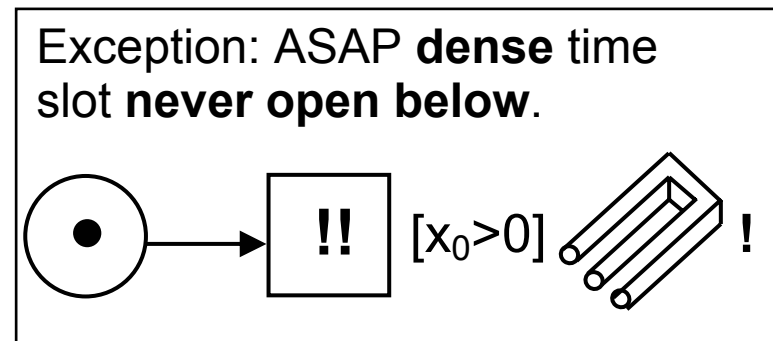
- the **natural** numbers \mathbb{N}_0 (or **integers** \mathbb{Z} , for differences).

Timer Nets - formally

A **timer Petri net** is an 9-tuple $N = (P, T, F, W, M_0, Type, X, C, ReSt)$, where

- $N = (P, T, F, W, M_0)$ is a PT-system,
- $Type$ is a mapping from T to $\{MAY, MUST, ASAP\}$,
- $X = \{x_0, x_1, \dots, x_r\}$ is a non-empty set of timer names,
- C is a mapping from T to the set $\Phi(X)$ of **timing conditions**, i.e. the language defined by

- $tc = true \mid x_i Rc \mid (x_i - x_j) Rc \mid tc \wedge tc,$
- $R = \langle \mid \leq \mid = \mid \geq \mid \rangle,$
- $x_i \in X,$
- $C \in \mathbb{T}.$



- $ReSt$ is a mapping $T \rightarrow \mathbf{P}(X \setminus \{x_0\})$, associating to each transition its **restart set** (set of restated clocks).

The dynamic behaviour of Timer nets

System start

The **system starts** to run

- at a physical time $x_0 = 0$;
Outside of the net, a real world moment must be associated with $x_0 = 0$.
- at the same moment, all (virtual) clocks in X begin to run, starting with a value of 0.
Now, all clocks run at the same speed, $\dot{x}_i = 1$.

The dynamic behaviour of Timer nets

Transition activation and occurrence

In order to be **temporally activated** (not only token-wise), a transition

- (1) must be **token-enabled** in the conventional sense,
i.e. by the presence of sufficiently many tokens on its input places, and
- (2) its **time condition** must be **fulfilled** (evaluate to TRUE)
with the clock readings given at this moment.

An activated transition **can occur**,

- (1) changing the marking as in a PT-system,
- (2) restarting all clocks in $ReSt(t)$ with value 0,
- (3) and doing all this without consuming time.

The dynamic behaviour of Timer nets

Passage of time

Time **can pass arbitrarily**,

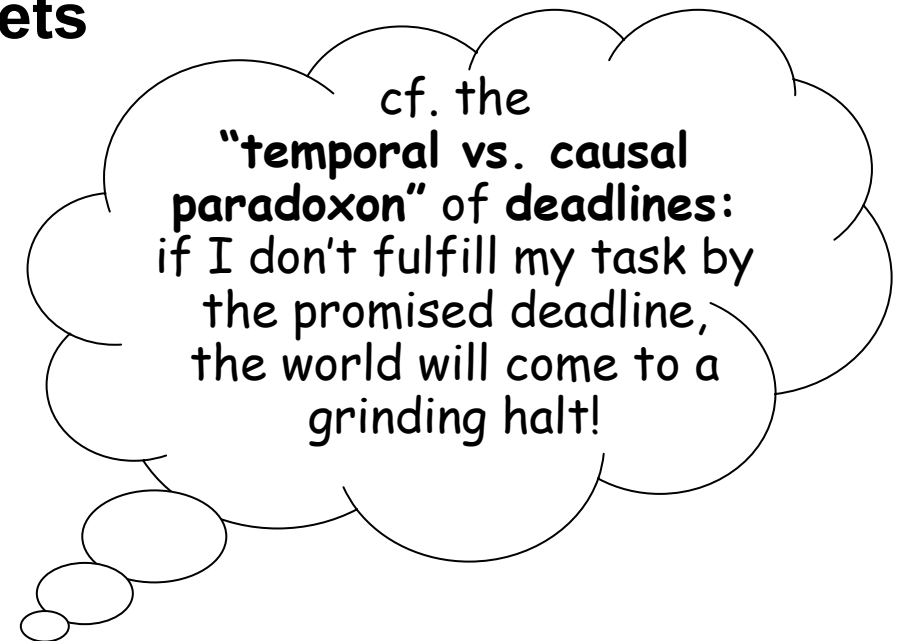
- before the first transition occurrence,
- between transition occurrences, and
- after the last transition occurrence,

but ... →

Time **“cannot pass”** if a **transition must occur**, i.e.

- an activated **MUST**-transition has reached its deadline, or
- an **ASAP** transition is activated.

In this case, one of any such transitions must occur right away,
and so on,
as long as there are any left.



The dynamic behaviour of Timer nets

Timed observations

The resulting **timed occurrence sequences** can be written as

$(t_1, v_1) (t_2, v_2) \dots (t_n, v_n) (\text{end}, v_{n+1})$, where

- v_i are clock-readings of x_0 , $v_{i+1} \geq v_i$, and
- no transition occurs between $x_0 = 0$ and v_1 as well as between $x_0 = v_n$ and v_{n+1} .

Why the final passage of time?

To characterize the **violation of deadlines!**

Non-timed models make it hard to deal with inactivity.

Exercise: Write timed traces of the previous examples and of the “Mall shop“!

Timing condition FAQ's

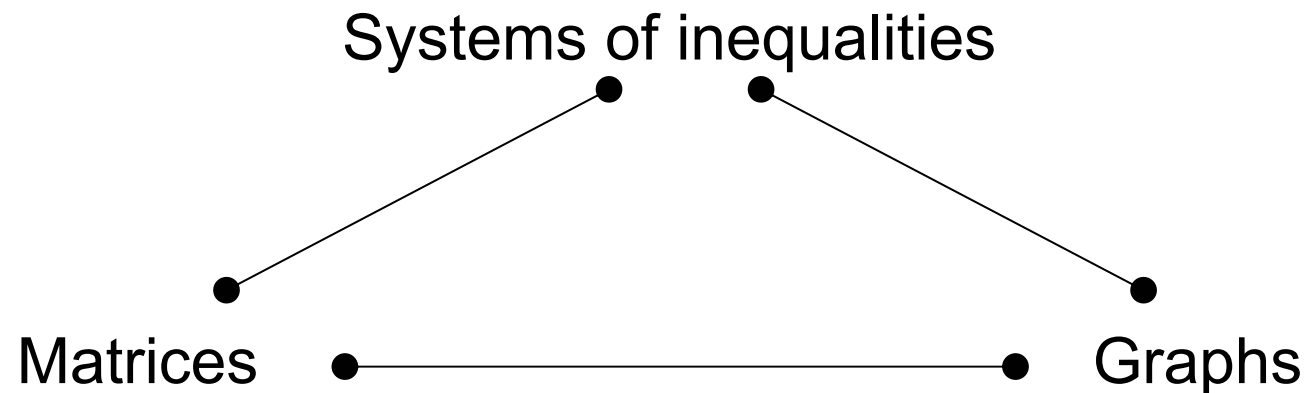
When are two timing conditions **equivalent**?
When is a timing condition **contradictory** (unfulfillable)?

Tools: simplified linear optimisation

Lucky correspondences



due to the special form
of the timing conditions!



System of inequalities → matrix

Let tc be a **timing condition** for $X = \{x_1, \dots, x_n\}$.

1. Transform to: $a \leq x_m - x_l \leq b, \quad l > m \rightarrow -b \leq x_l - x_m \leq -a$

2. Add conds.: $0 \leq x_k - x_k \leq 0 \quad (1 \leq k \leq n), \quad -\infty \leq x_i - x_k \leq \infty \quad (1 \leq i, k \leq n).$

3. Form $\left\{ \begin{array}{l} \text{Maximum of all lower bounds} \\ \text{Minimum of all upper bounds} \end{array} \right\}$ of $\left\{ \begin{array}{l} x_k \\ x_i - x_k, \quad 1 \leq k \leq i \leq n \end{array} \right\}$

4. → **Interval form:**

$$\forall 1 \leq k \leq n: \quad -D_{0k} \leq x_k \leq D_{k0}$$

$$\forall 1 \leq k \leq i \leq n: \quad -D_{ki} \leq x_i - x_k \leq D_{ik}$$

or ... with $x_0 \equiv 0$

→ **Upper bound form**,: $\forall 1 \leq k, i \leq n: x_i - x_k \leq D_{ik}$

or ...

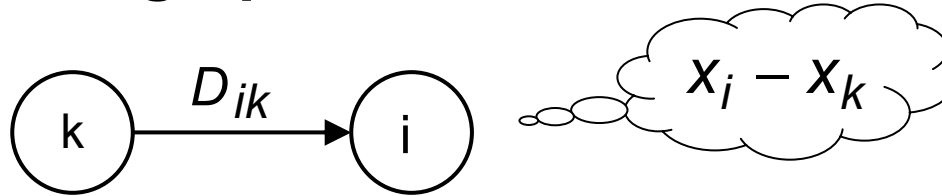
5. → **Difference Bound Matrix, DBM**, for tc

$$D(tc) = (D_{ik}) \quad i, k = 0, \dots, n$$

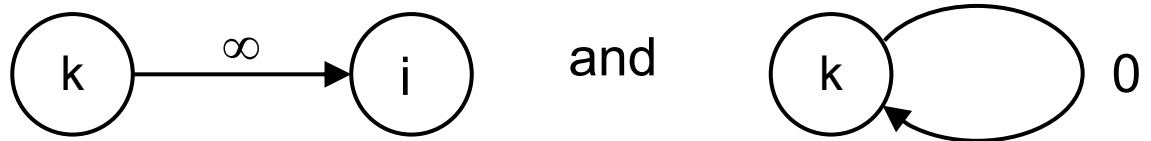
**Normal-
form(s)**

From the matrix to the cost graph

Schema:

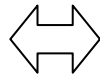
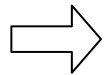


omissible as default:

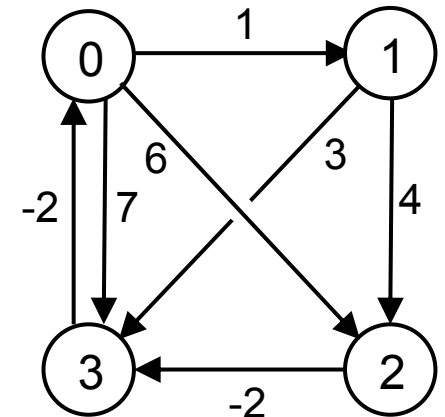
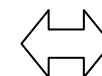


Example:

$$\begin{array}{l}
 x_1 \leq 1 \\
 x_2 \leq 6 \\
 2 \leq x_3 \leq 7 \\
 x_3 - x_1 \leq 3 \\
 x_2 - x_1 \leq 4 \\
 2 \leq x_2 - x_3 \\
 x_0 - x_0 \leq 0 \\
 x_0 - x_1 \leq \infty \\
 \vdots \\
 x_3 - x_1 \leq 3 \\
 x_3 - x_2 \leq -2 \\
 x_3 - x_3 \leq 0
 \end{array}$$



$$\begin{pmatrix}
 0 & \infty & \infty & -2 \\
 1 & 0 & \infty & \infty \\
 6 & 4 & 0 & \infty \\
 7 & 3 & -2 & 0
 \end{pmatrix}$$



Canonical representations of timing conditions

Some inequalities imply others: $(x_2 - x_1 \leq 4) \wedge (x_1 \leq 1) \Rightarrow (x_2 \leq 5)$

→ In timing condition: sharpened inequalities
(still same solution space!)

in DBM: $D_{20} \leq D_{21} + D_{10}$

in graphs: „cheaper“ paths.

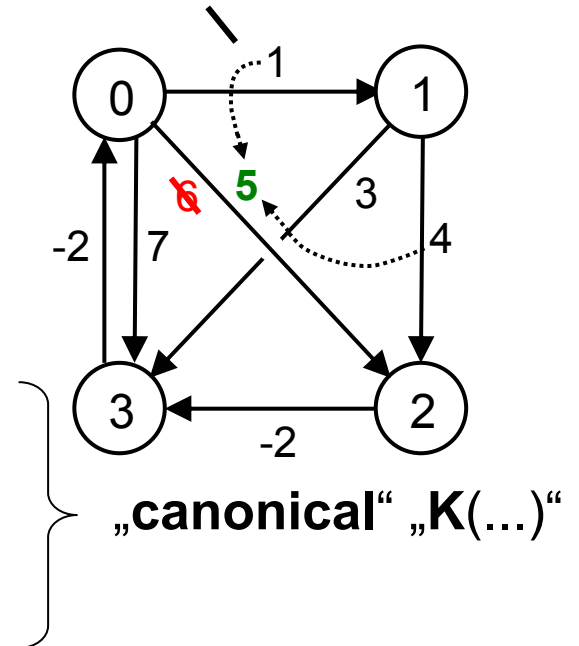


sharpest inequalities

$\leftrightarrow \forall 0 \leq i, j, k \leq n : D_{ik} \leq D_{ij} + D_{jk}$ in $D(tc)$

\leftrightarrow **minimal cost graph:** $G(tc)$

in tc



„canonical“ „K(...)"

tc_1, tc_2 **solution equivalent**

$\Leftrightarrow K(D(tc_1)) = K(D(tc_2))$

timer condition tc ist **fulfillable**

$\Leftrightarrow K(D(tc_1))$ does not contain $-\infty$

$\Leftrightarrow G(tc)$ does not contain „winning cycle“

Floyd-Warshall-Algorithm

computes →

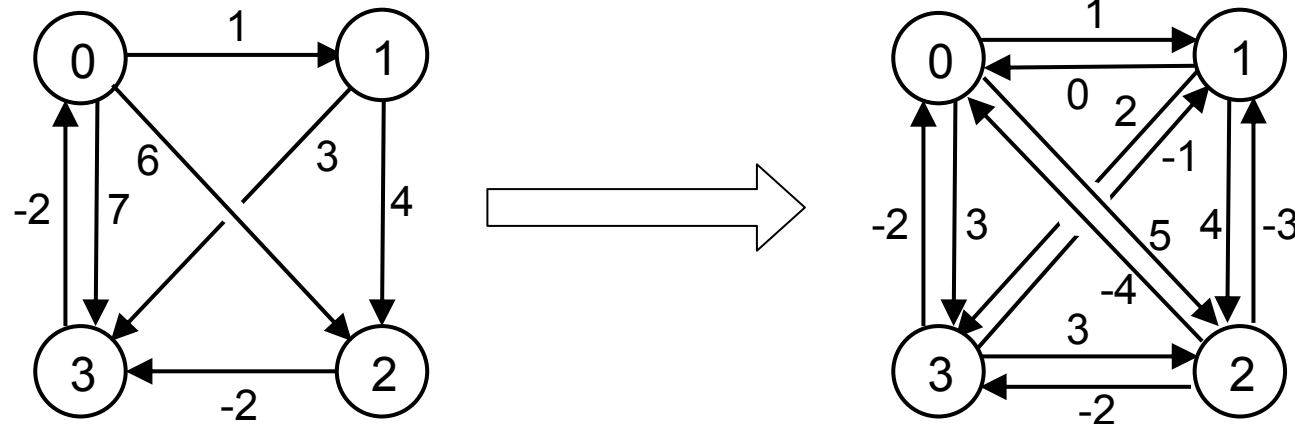
minimal cost graph for a given digraph with real valued edge “costs”

hence also a canonical DBM and a canonical normal form of a timing condition.

```

FOR j:=1,...,n
  FOR i:=1,...,n
    FOR k:=1,...,n
       $D_{ik}^{(j)} := \min(D_{ik}^{(j-1)}, D_{ij}^{(j-1)} + D_{jk}^{(j-1)})$ 
    
```

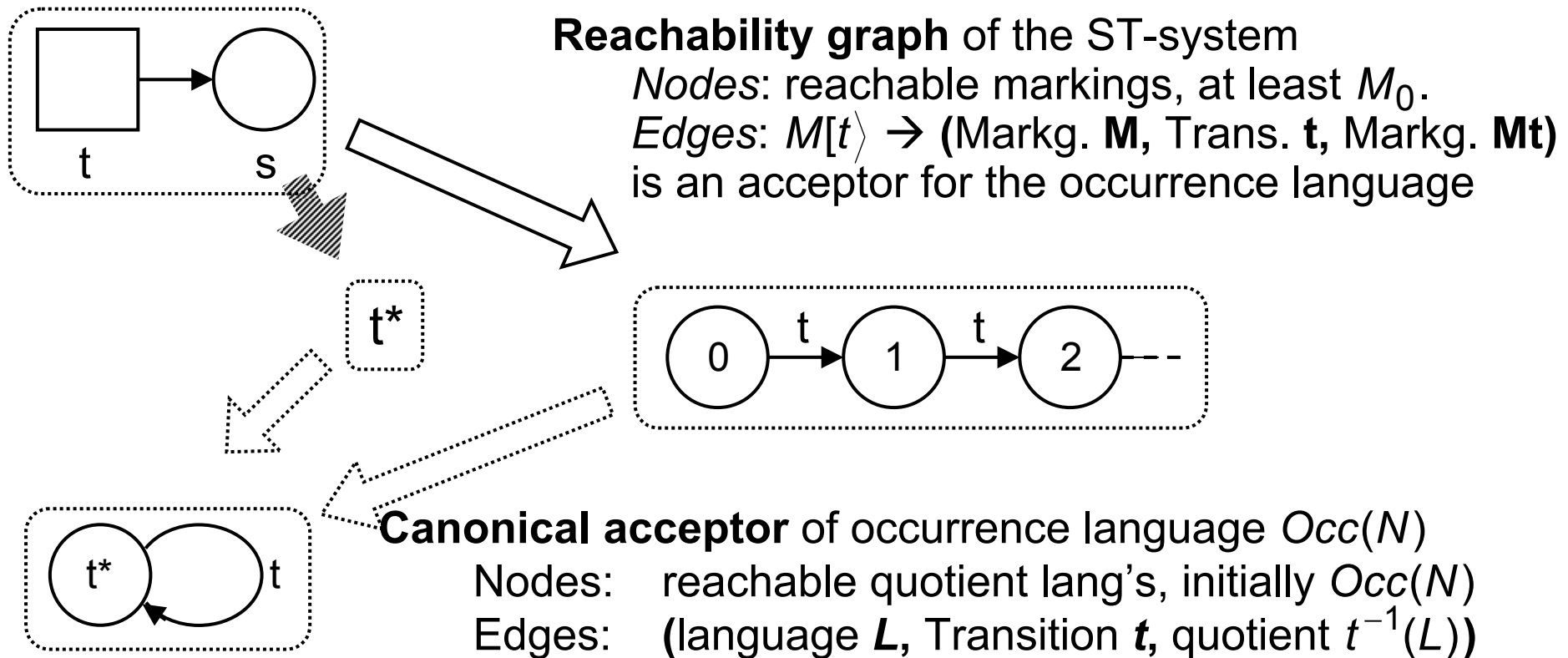
In our example:



Reminder: untimed acceptors

Aim:

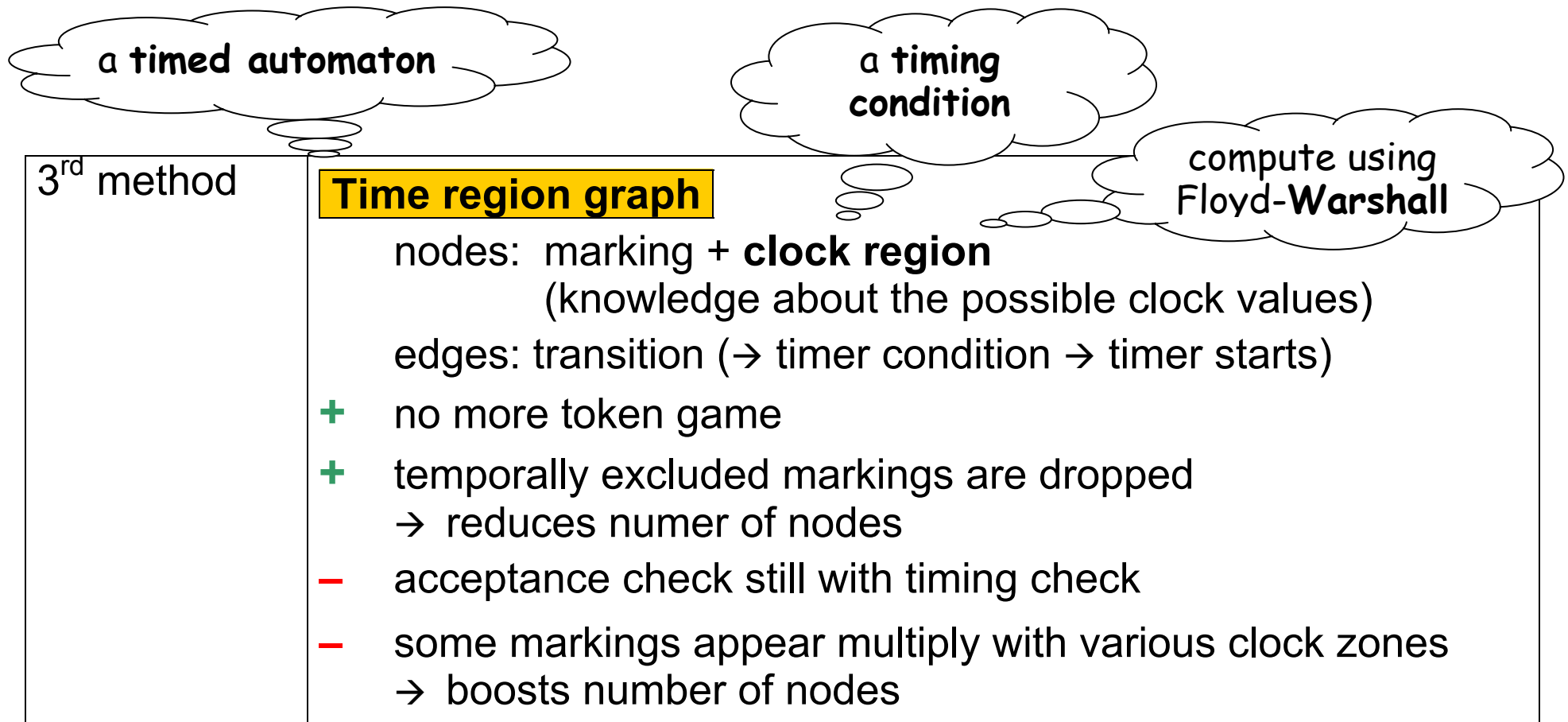
simple data object allowing to decide in a simple manner whether an observation sequence is valid for the specification (supports conformance).



“Acceptors” for timed observation sequences (1)

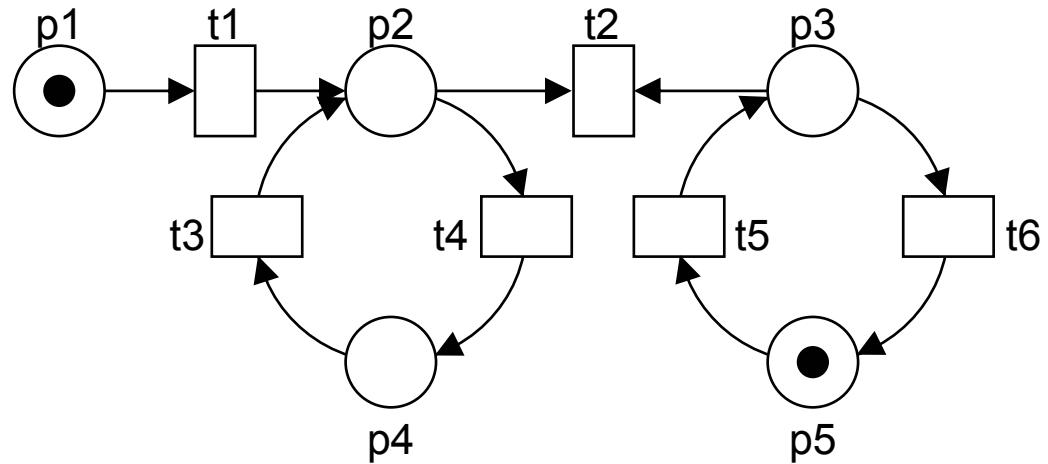
1 st method	Timer net itself recipe: replay the sequence to be investigated → clumsy : token game + checking timing conditions → problematic : non-deterministic if labelled net (which copy with this name fired?)
2 nd method	Timed reachability graph nodes: marking + time x0 edges: timed occurrence/ passage of time + no more token game - usually infinite graph

“Acceptors” for timed observation sequences (2)

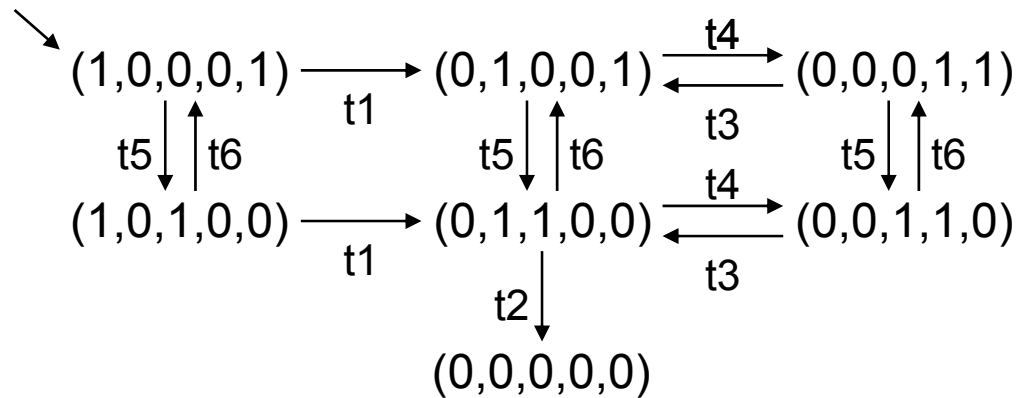


Time region graph, a partial example – UNTIMED NET

PT system



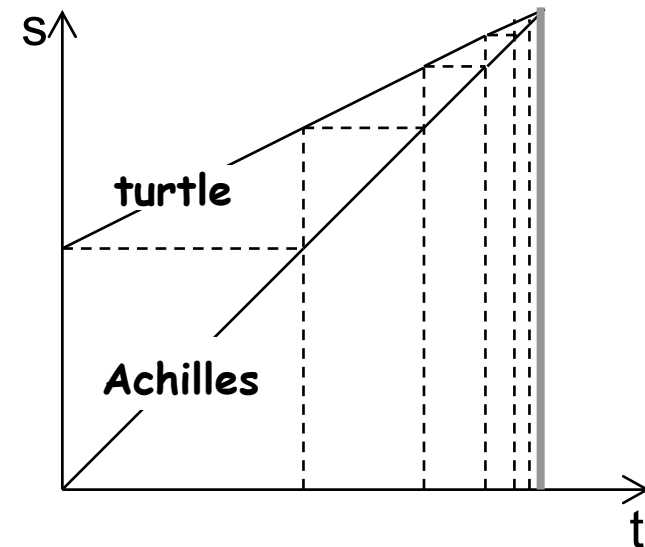
Reachability graph



Zenon effects

A **timed system specification** $Spec$
(from a **class** C of such specifications)
has the **Zenon property** if it allows an
infinite observation sequence within a **bounded time interval**.

Zenon paradoxon (one of several of his):



Typical questions:

- is Zenon pr. **possible** within C , or is C **non-Zenon**?
- is this **decidable** for C , and, if yes, **how**, and how **complex**?


 Probably

Weaker precursors of Timer nets (1)

Time Nets (Merlin, Farber 1976) – **Waiting time nets**

6-tuple (P, T, F, M_0, E, L) , where (P, T, F, M_0) ST-system and

$E : T \rightarrow \mathbb{N}_0$ **earliest firing time** of transition

$L : T \rightarrow \mathbb{N}_0 \cup \{\infty\}$ **latest firing time** of transition

E = uninterruptedly **token-activated waiting time**

Preemption:

deactivation by a transition $u \neq t$, while waiting, is possible.

L = like **deadline** for **MUST** transitions

Start: Initial marking, and “system clock” starts at 0.

natural semantics: sequences (transition, moment of firing) (...) (...) . . .

Probably

Weaker precursors of Timer nets (2)

Timed Nets (Ramchandani 1974) – **Reaction time nets**

5-tuple (P, T, F, M_0, D) , where (P, T, F, M_0) ST-system and

$D : T \rightarrow \mathbb{R}_{\geq 0}$ **Firing duration** of transition

Firing mechanics:

Once token-activated (and winner of possible “**token competition**”) ...

Start: Input tokens are collected \rightarrow **busy-state**

After $D(t)$ time units:

End: Output tokens are dispensed \rightarrow **idle-state**

[dialects with or w/o self concurrency]

natural semantics? similar, but

Do we **observe** firing **starts**, **ends**, or **both**?

A widely known contemporary of Timer nets

Timed Automata (Alur, Dill 1990-1994)

6-tuple $A = (L, L_0, \Sigma, X, I, E)$, where

- L finite set of **locations;**
- $L_0 \subseteq L$ **initial locations;**
- Σ finite set of **labels;**
- X finite set of **clocks;**
- $I: L \rightarrow \Phi(X)$ **“invariants”;** $\left(\begin{array}{c} \text{timing} \\ \text{conditions} \end{array} \right)$
- $E \subseteq L \times \Sigma \times \Phi(X) \times \mathbf{P}(X) \times L$ **transitions** (firings).

Time passes while the timed automaton is in a location.

Probably

Stronger successors of Timer nets

AN-Timed nets (Abdullah, Nylén, 2001. Cf. Lautenbach et al.: Timestamp nets, 1998)

- Each **token** has an **age** which is represented by a real-valued clock

A marking is thus a multiset of real numbers (the ages of the tokens) on each place. These ages/numbers grow at the same speed.

- Input **arcs** of transitions are labelled by **timing conditions** (intervals) fixing the accepted ages of the **eligible** incoming tokens.

Tokens of a rejected age are neglected.

Activation is as usual but refers only to eligible tokens.

- All transitions are **MAY** (in Timer net terms): they do not have to fire ASAP, they can even let their activation expire (not MUST).
- Tokens produced start with their clock at 0.

Increased expressivity due to unlimited number of clocks?

Expressivity – a neglected topic

Many classes C of specifications

(e.g. timer nets, time nets, timed nets, AN-timed nets, timed automata)

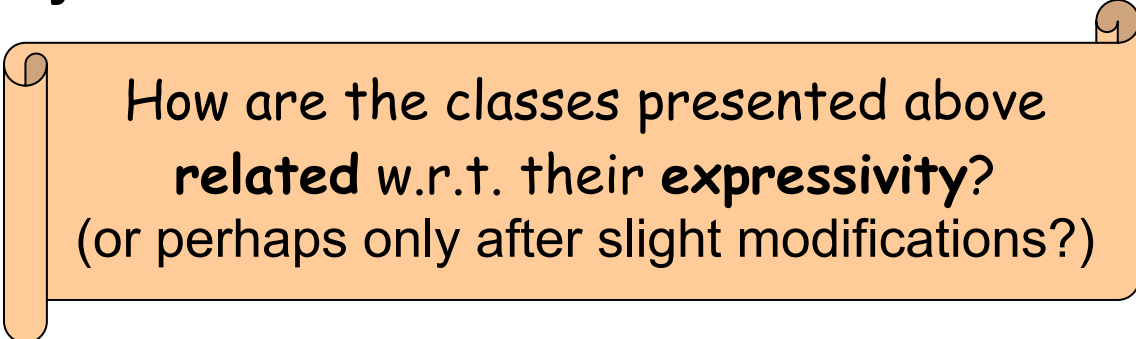
define a class **Languages(C)** of timed observation languages.

C_1 is **more expressive** than C_2 if

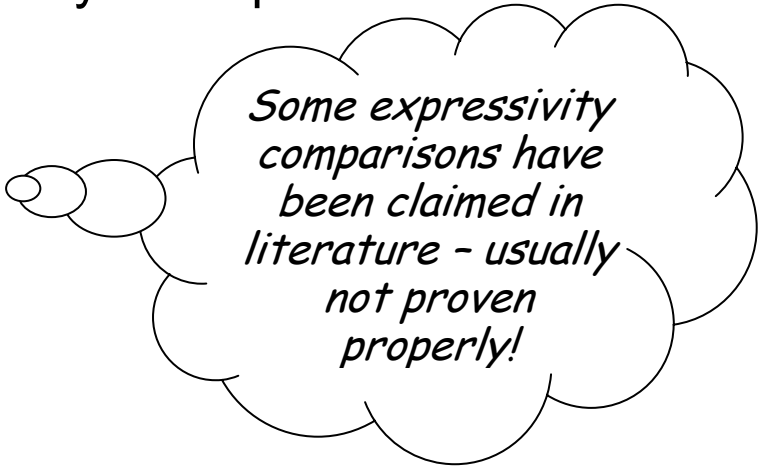
$$\text{Languages}(C_1) \supseteq \text{Languages}(C_2),$$

and **strictly more expressive**, if they are additionally not equal.

My Question:



How are the classes presented above
related w.r.t. their **expressivity**?
(or perhaps only after slight modifications?)



*Some expressivity
comparisons have
been claimed in
literature - usually
not proven
properly!*

Literature

Abdulla, P.A., and A.Nylén, Timed Petri Nets and BQOs, *Intl. Conf. on Application and Theory of Petri Nets 2001*, LNCS vol. 2075, pp. 53-70, Springer, 2001

R. Alur, D.L. Dill: A theory of timed automata, *Theoretical Computer Science*, 126:183--235, 1994.

Baumgarten, B., *Petri-Netze – Grundlagen und Anwendungen*, BI Wissenschaftsverlag, Mannheim, 1st ed., 1990 (first mentioning of Timer nets)

P. Merlin, D.J. Farber: Recoverability of communication protocols, *IEEE Trans. on Communications*, 24(9), pp. 1036-1043, 1976

C. Ramchandani: *Analysis of asynchronous concurrent systems by timed Petri nets*, Technical Report MAC-TR-120, MIT, 1974

J. Wang: *Timed Petri Nets – Theory and Application*, Kluwer Academic Publishers, 1998