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

High Level Petri Nets

Bernd Baumgarten

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

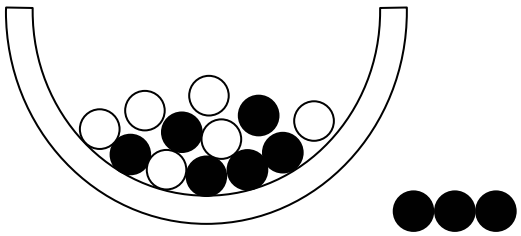
Introducing High Level Petri Nets

Black and white pebble "game"

Given: Pot with *m* **white** and *n* **black** pebbles + supply of ≥[*m*/2] black pebbles

1 Move: Take any two pebbles out of the pot.

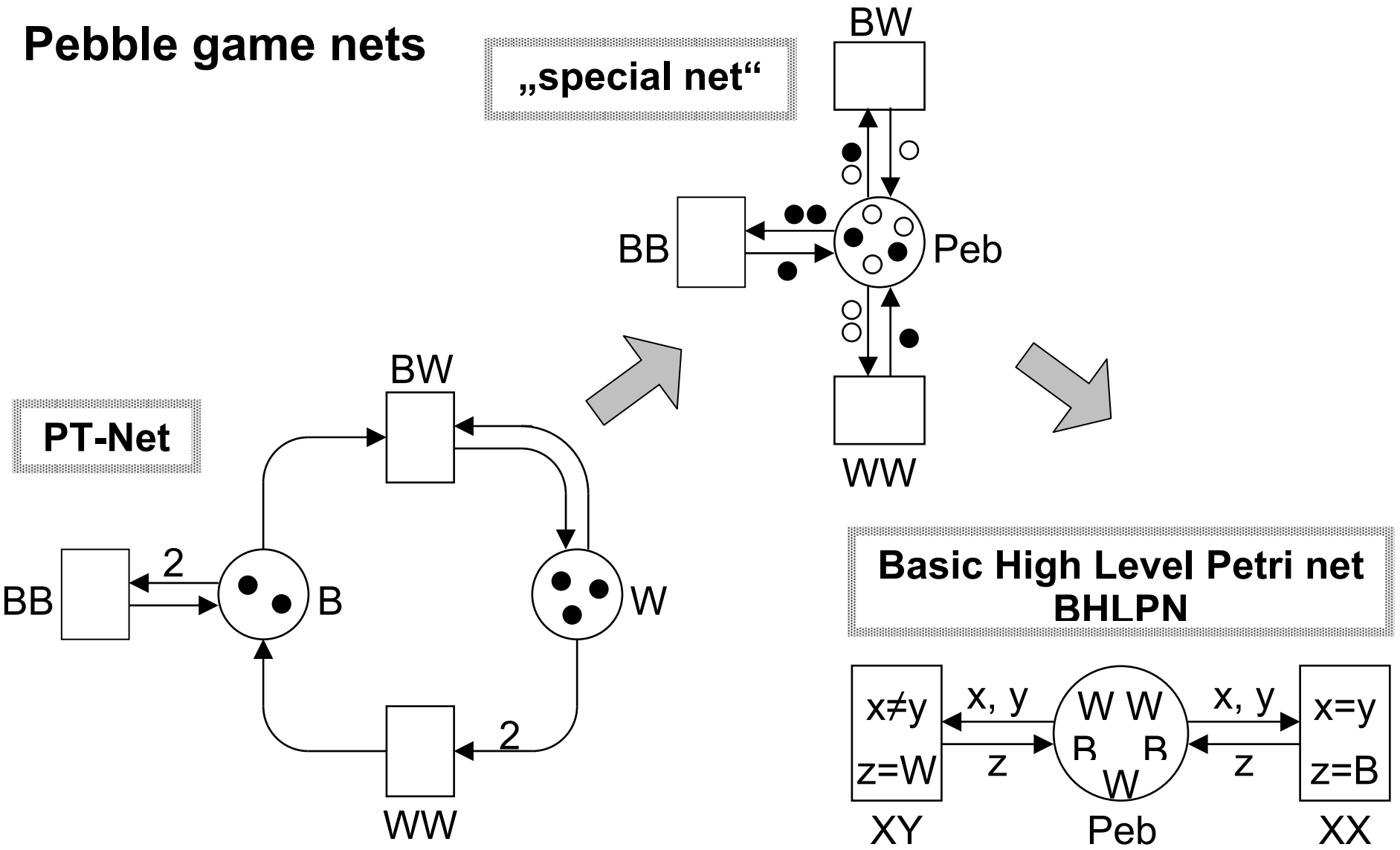
- B + W → put 1 W back.
- W + W → put 1 B into pot.
- B + B → put 1 B back.



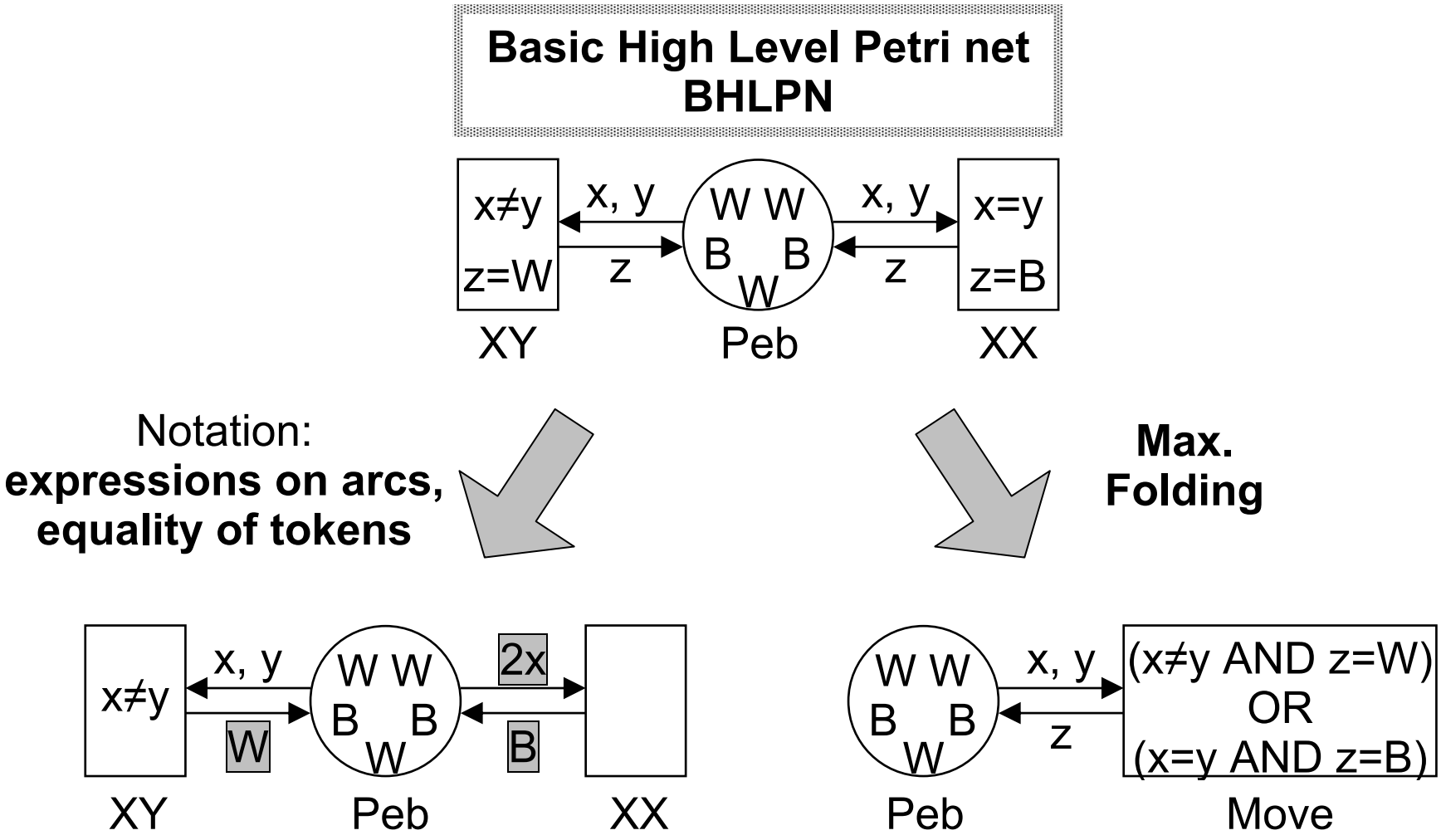
Questions: Initial situation (*m*,*n*) → **one** or **several** possible final situations?
If exactly one, *final*(*m*,*n*) ∈ $\mathbf{N}_0 \times \mathbf{N}_0$, **calculate** it!

Task: **Model** the system's behaviour and states.

Pebble game nets



More pebble game nets



Informal definition of High Level Petri Nets

PT systems:

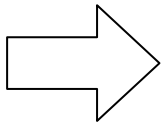
- tokens are **all alike** and cannot be distinguished

High level Petri nets:

- tokens are individual; they can be **distinguished**;
- one can **calculate** with them.

For various reasons
stylistic prefences,
various dis/advantages of various variants,
publish or perish,
many kinds of HLPNs have been defined.

Formal definitions of HLPNs are often very long.



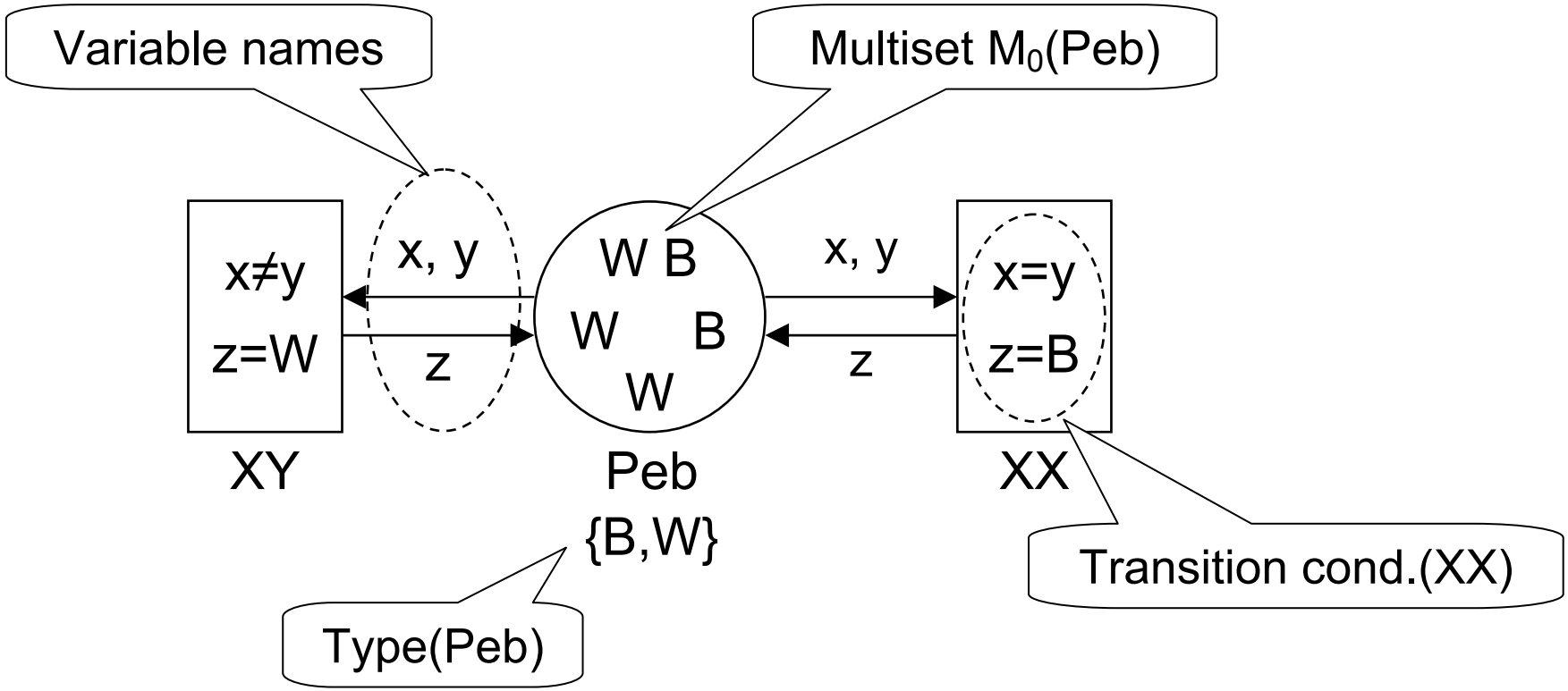
For the sake of simplicity, we introduce **informally** what we call **Basic HLPNs**.

Basic HLPNs

… differ from PT systems in the the following static features:

- Each **place** ↦ a data type *Type*(*p*) (token sort, class, kind).
- Each **marking** of a place = a multiset of individual tokens (of the type assigned to the place).
- Each **arc weight** of an arc = a list of variable names all different around any transition (can be referred to in the transition conditions).
- Each **transition** ↦ a transition condition *Cond*(*t*) (requirements on input and output tokens).

BHLPN features



HLPN separate declaration part

- declare **concrete data types**
  (sorts, constants, operations, properties, relations),

  e.g. as provided by a **programming language** or a dedicated **declaration part language**

- or declare an **abstract data type**

  in some **ADT dialect**

- assign token **types/sorts** to **places** of the net

Dynamic features of Basic HLPNs

The **occurrence** of a transition *t* is determined by an

- **assignment *ass*** of values to the variables appearing in the arc weights around *t*.

transition occurrence = (transition name, assignment)

This assignment associates

- to each arc weight *W*(*f*) (= variable list) concerned
- a **multiset** of values ***ass_w*(*W*(*f*))** .

The occurrence of *t* with this assignment must fulfil the following conditions:

- For each input arc (*p*,*t*), the multiset *ass_w* (*W*(*p*,*t*)) must be contained in the given marking *M*(*p*) of the input place.
- The assignment must make *ass_C*(*Cond*(*t*)), the transition condition of *t,* true.

The occurrence(s) of an HLPN transition

This occurrence of *t* has the following effects:

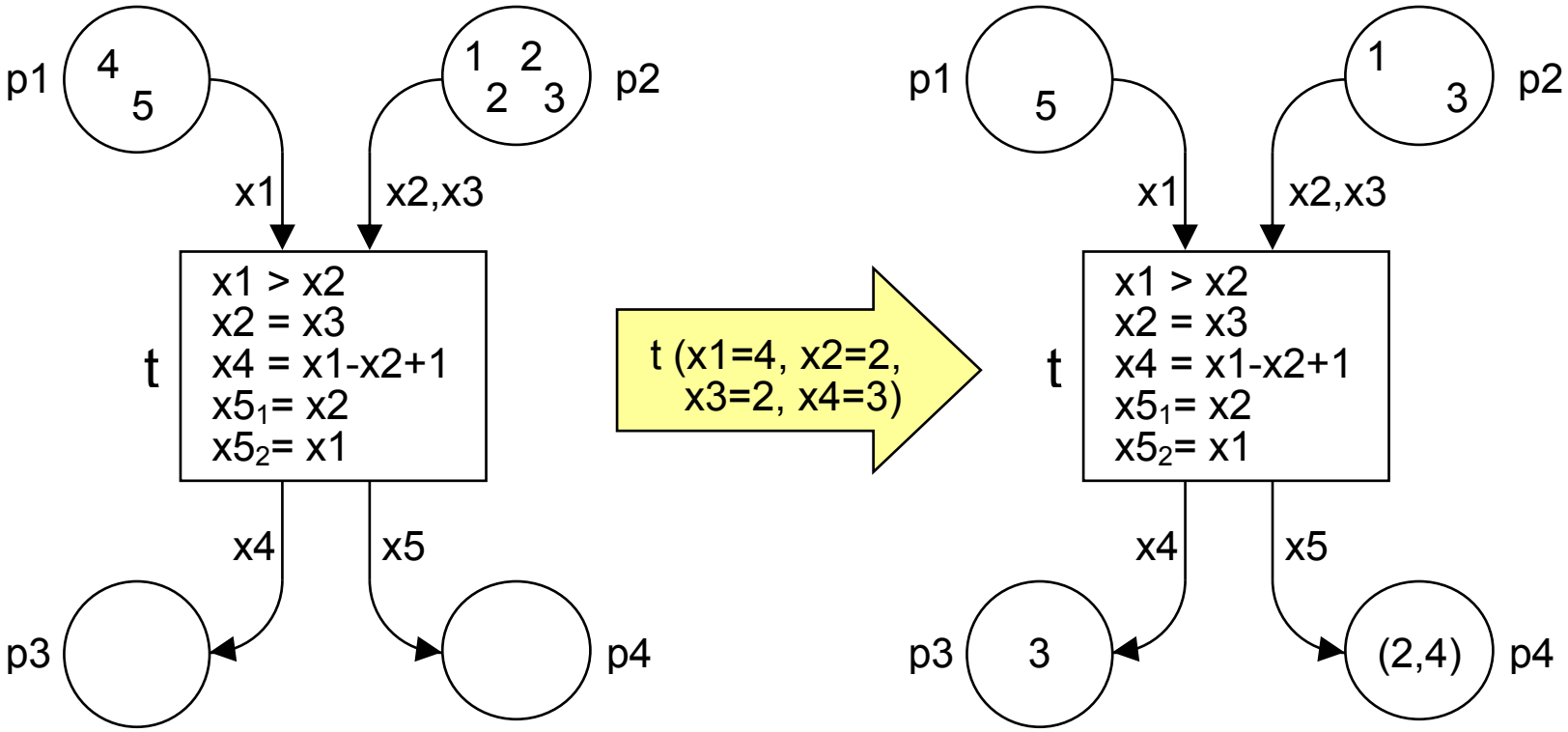
- For each input arc (*p*,*t*),
  the multiset ***ass*(*W*(*p*,*t*))** is **taken away** (subtracted) from *M*(*p*).

- For each output arc (*t*,*p*),
  the multiset ***ass*(*W*(*t*,*p*))** is **added** to (whatever is left of) *M*(*p*).

All this happens atomically, without (noticeable) intermediate state.

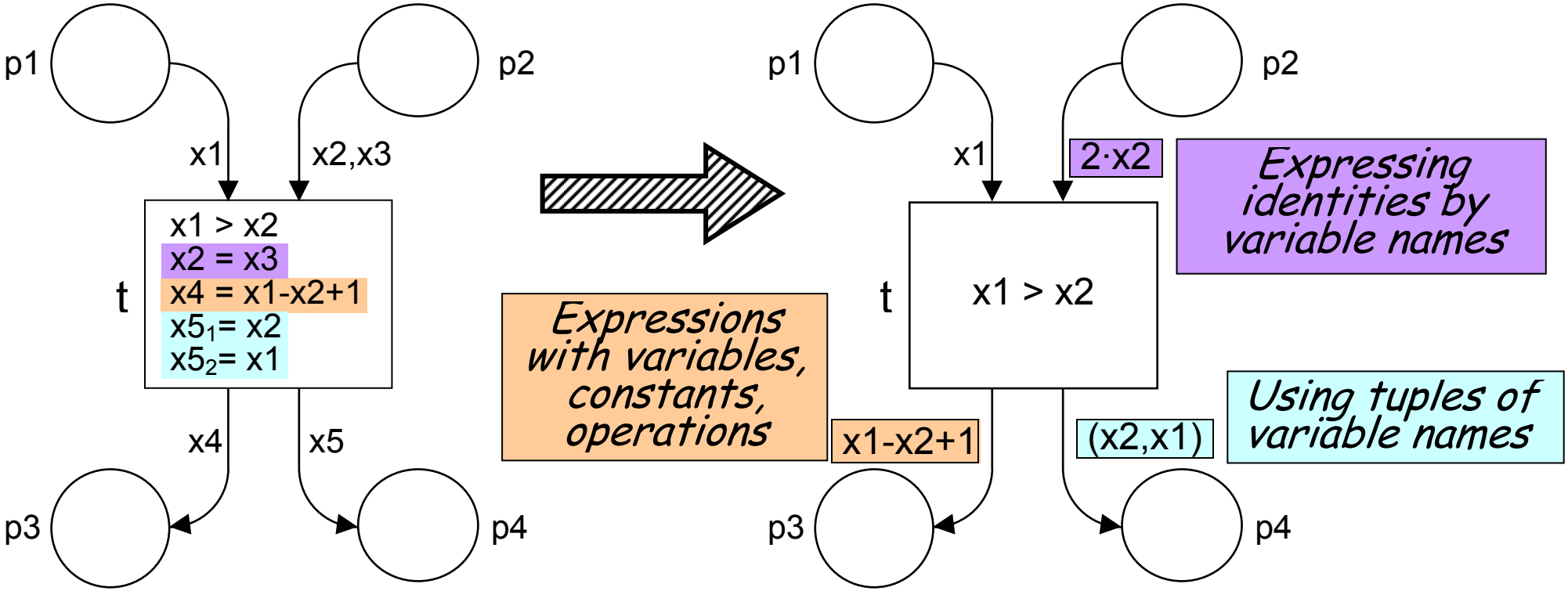
The occurrence(s) of an HLPN transition, example

p1, p2, p3 are of type IN. p4 is of type IN x IN.

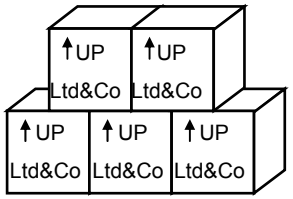


t above can also fire a **different occurrence**
with different parameter (occurrence variable) **values**.

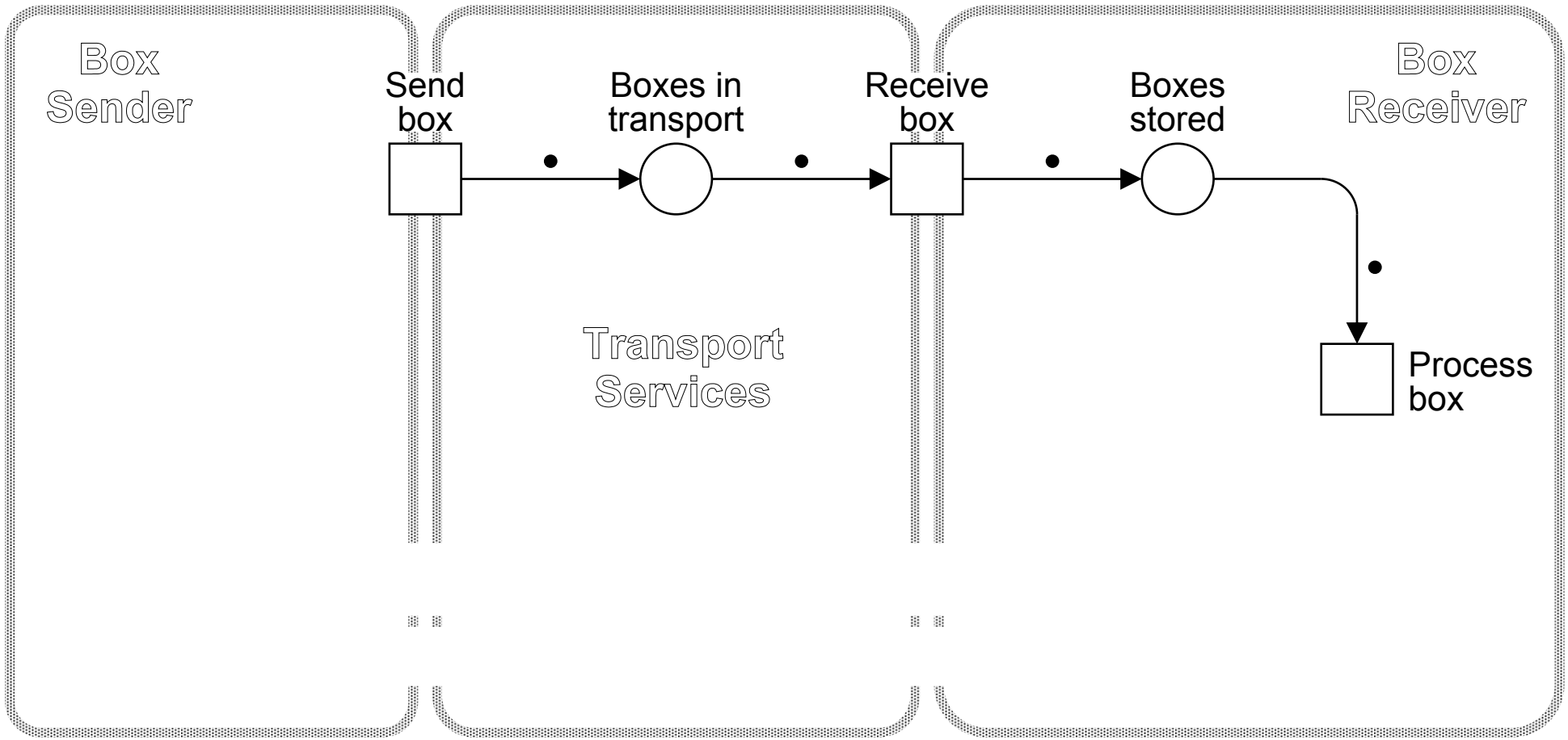
Notational simplifications:



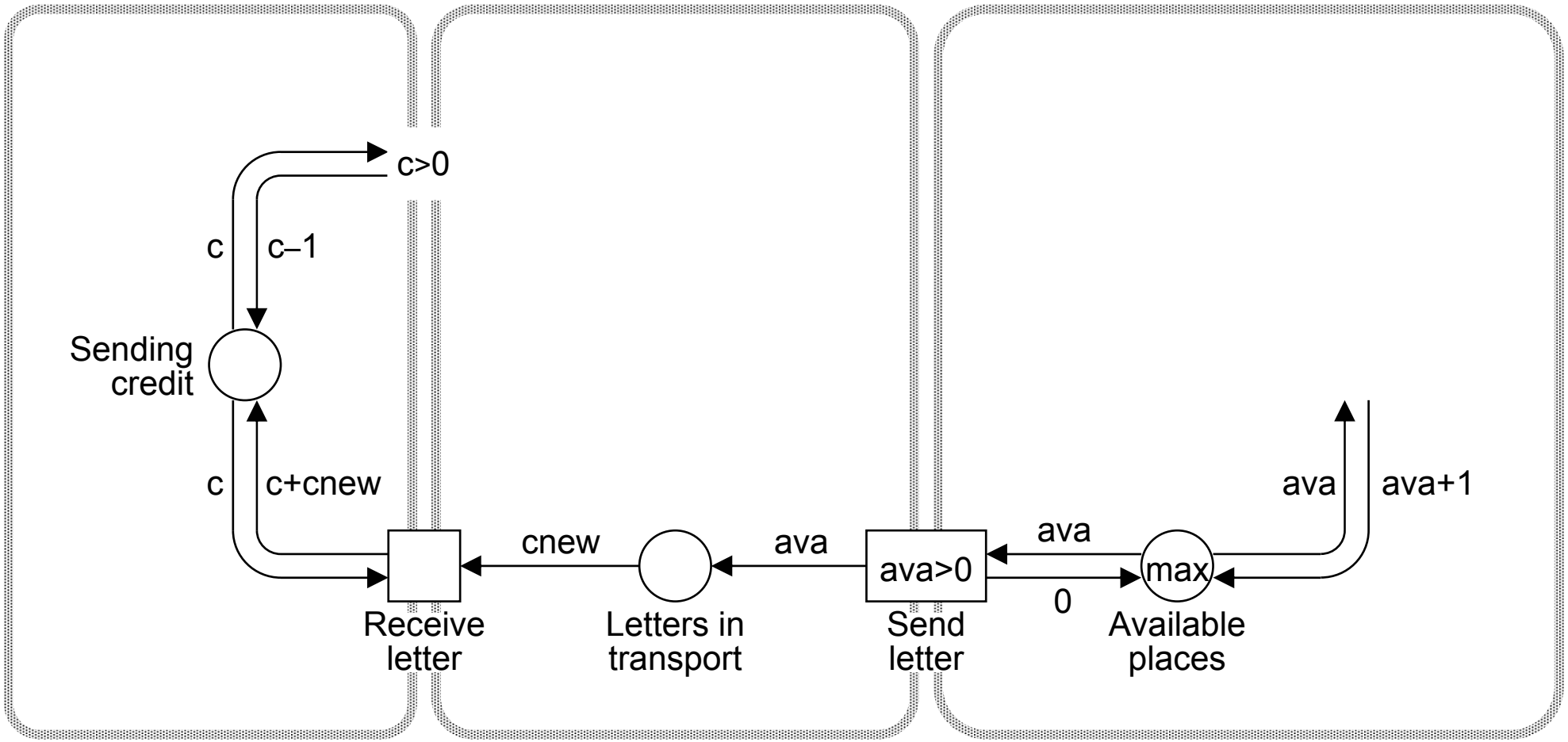
Working with HLPNs: a flow control example



A flow-control problem: limited storage space for boxes but reception cannot be blocked

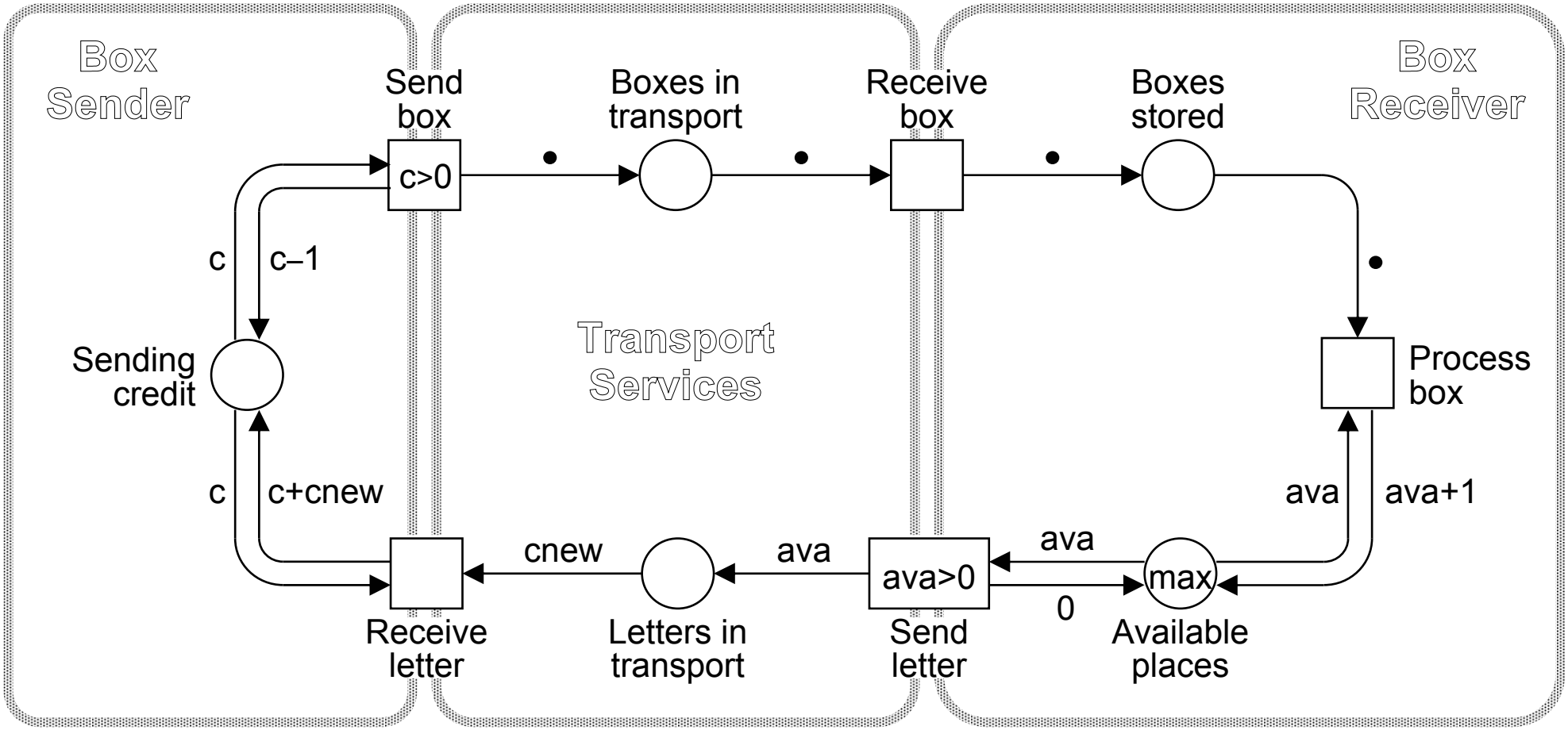


Working with HLPNs: a flow control example



A flow-control solution: the "boxes and letters protocol"

Correctness ←→ an invariant w.r.t. transition ocurrences



nr in *ava pl* + Σ(nrs in *letters*) + nr in *send cred* + # tokens in *boxes in tp*
+ #tokens in *boxes stored* = constant = **max**