Property preserving transformations: State space and structure reduction

dr. Tamás Bartha

dr. István Majzik

dr. András Pataricza

BME Department of Measurement and Information Systems

State space reduction: Partial order reduction

Simplification of the reachability problem

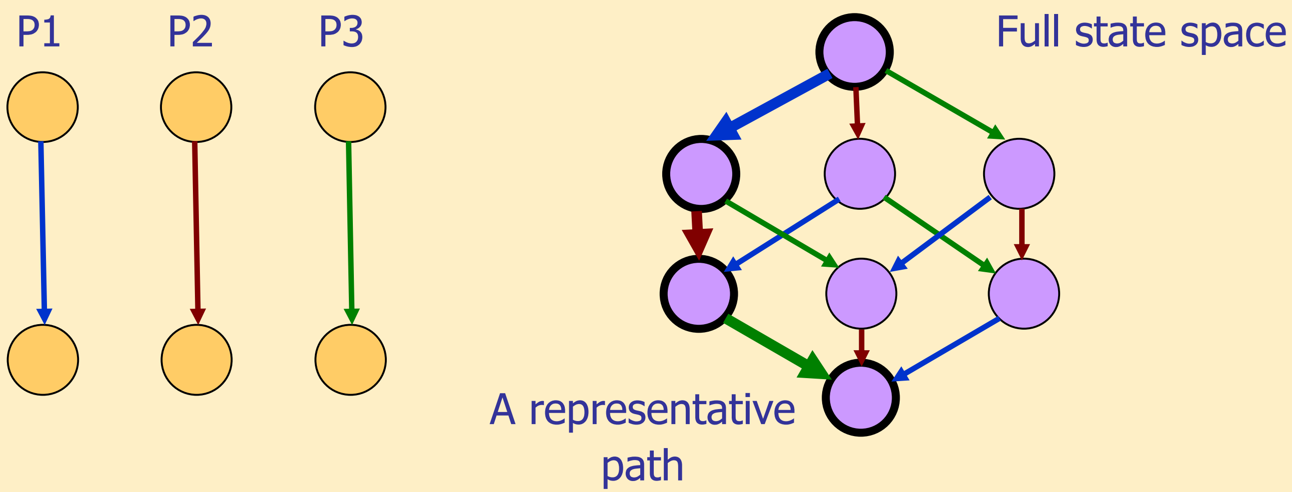
- Reduction while preserving the selected properties
  - Expressive power of the model decreases (non-selected properties may become modified or lost!)
  - Functionality changes, but the changes are controlled
  - The new model represents ("covers") the original one regarding the selected properties
  - Many kinds of property preserving transformations exist

Ideas for simplifying the reachability problem

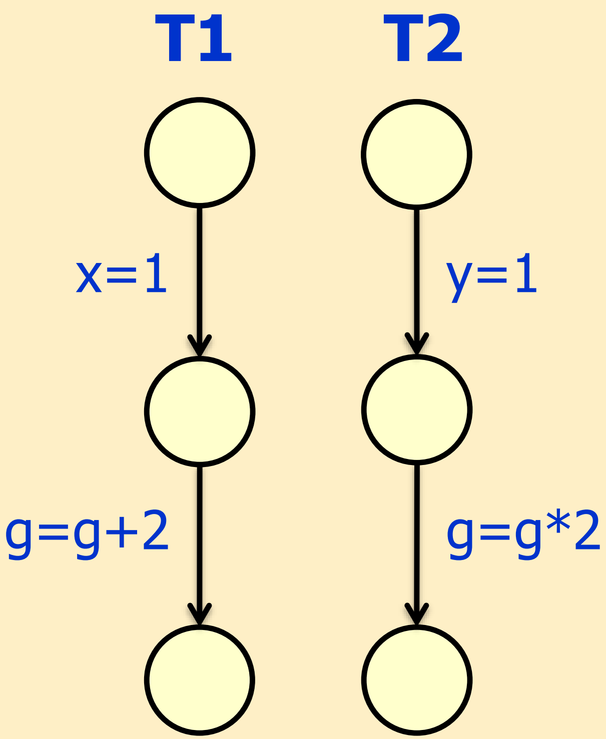
- Exploiting symmetries
  - Examine identical network parts only once
    - E.g. resource groups: components with the same behavior
  - Invariance for cyclic permutation
    - Colored Petri nets $\rightarrow$ Well-formed colored Petri nets (WFN) (see later!)
- Increasing the efficiency of state space traversal
  - Traverse only states “of interest”
    - Property preserving reduction
  - Traverse only necessary transitions
    - Omit alternative paths

A possible reduction: Partial order

- Reachable states form a partially ordered set
- Asynchronous behavior: overlapping → alternative paths with the same results
- Alternative paths are redundant regarding the final state (reachability); traversal of a single representative path may be sufficient



Example: Execution of alternative paths

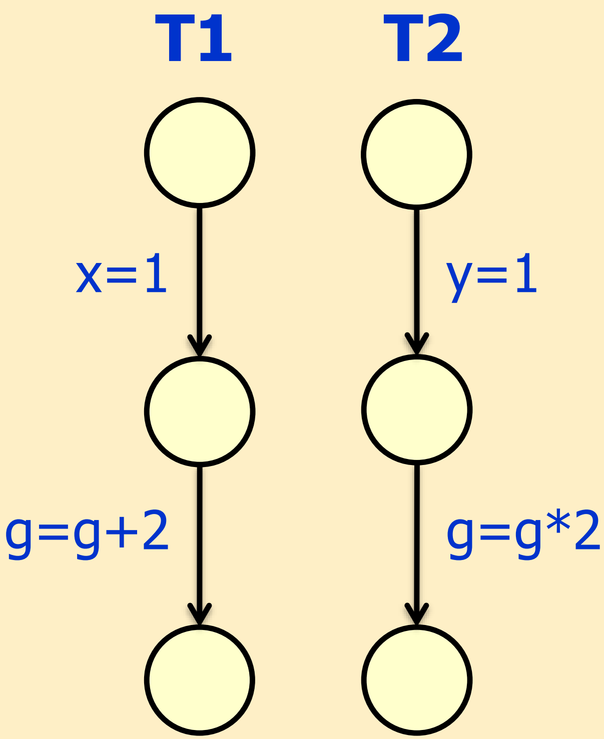


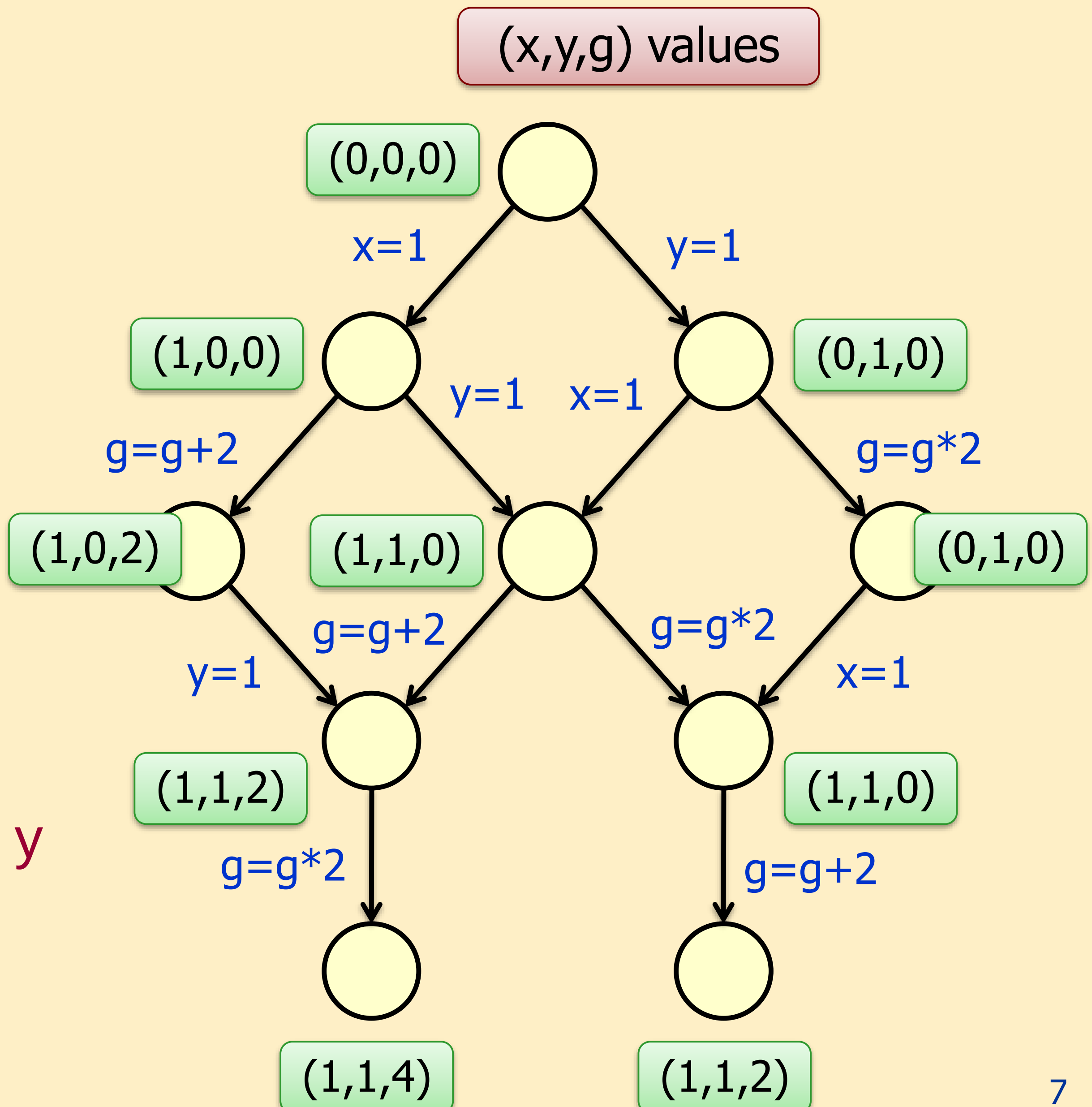
- Local variables:

  x and y
- Global variable:

  g
- 6 possible executions:
  1. x=1; g=g+2; y=1; g=g*2
  2. x=1; y=1; g=g+2; g=g*2
  3. x=1; y=1; g=g*2; g=g+2
  4. y=1; g=g*2; x=1; g=g+2
  5. y=1; x=1; g=g*2; g=g+2
  6. y=1; x=1; g=g+2; g=g*2

Example: Alternative states and paths

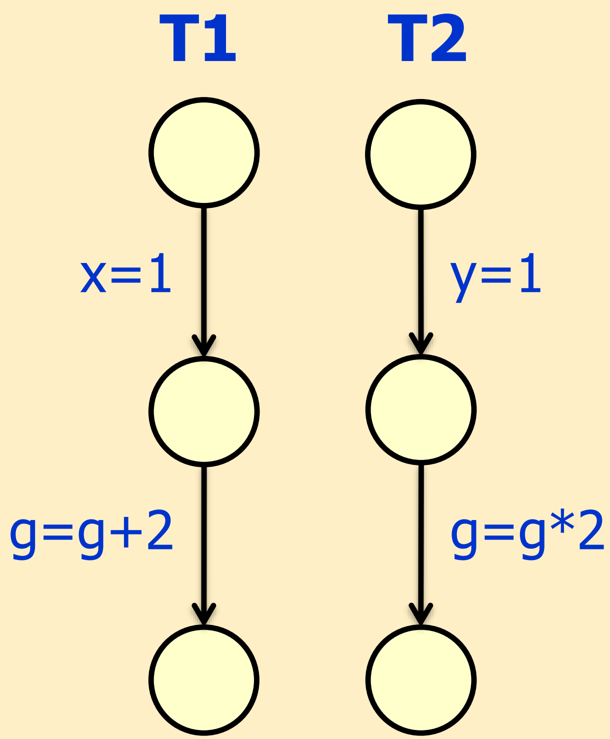




Local variables: x and y

Global variables: g

Example: Dependencies



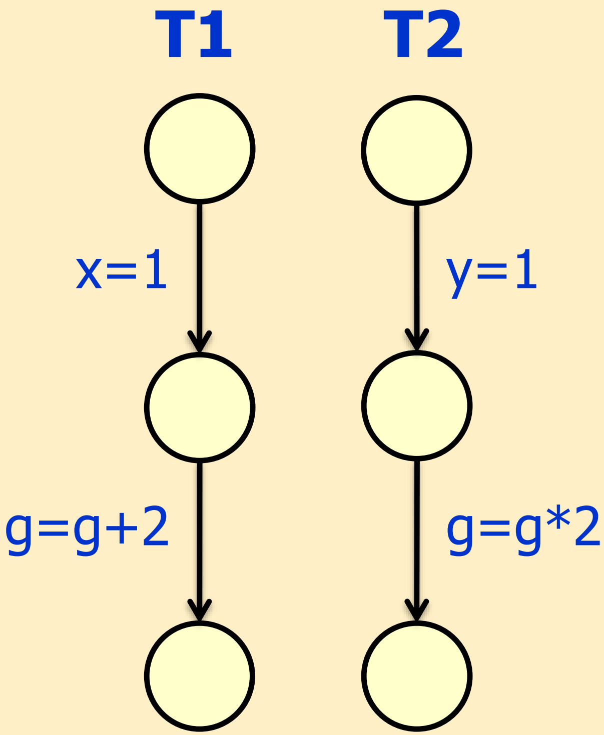
	x=1	y=1	g=g+2	g=g*2
x=1		I	C	I
y=1	I		I	C
g=g+2	C	I		D
g=g*2	I	C	D	

I: Independent

C: Control dependency

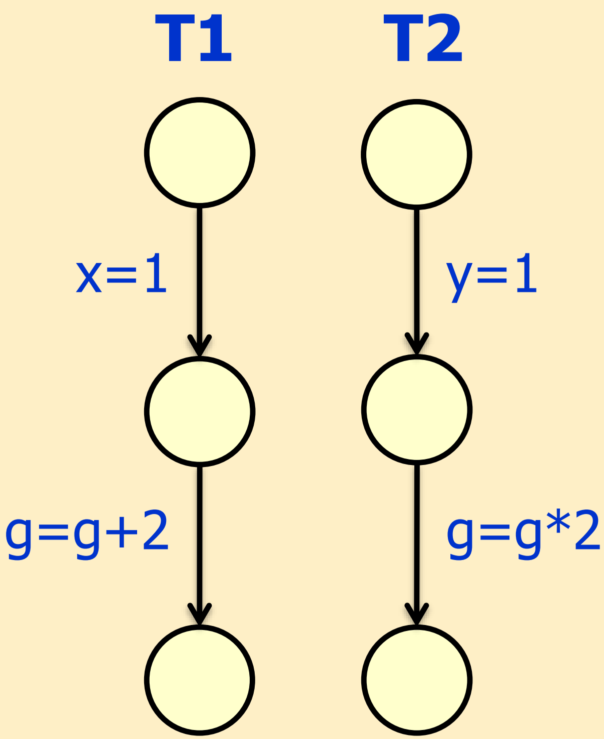
D: Data dependency
(using common variables: different order → different result)

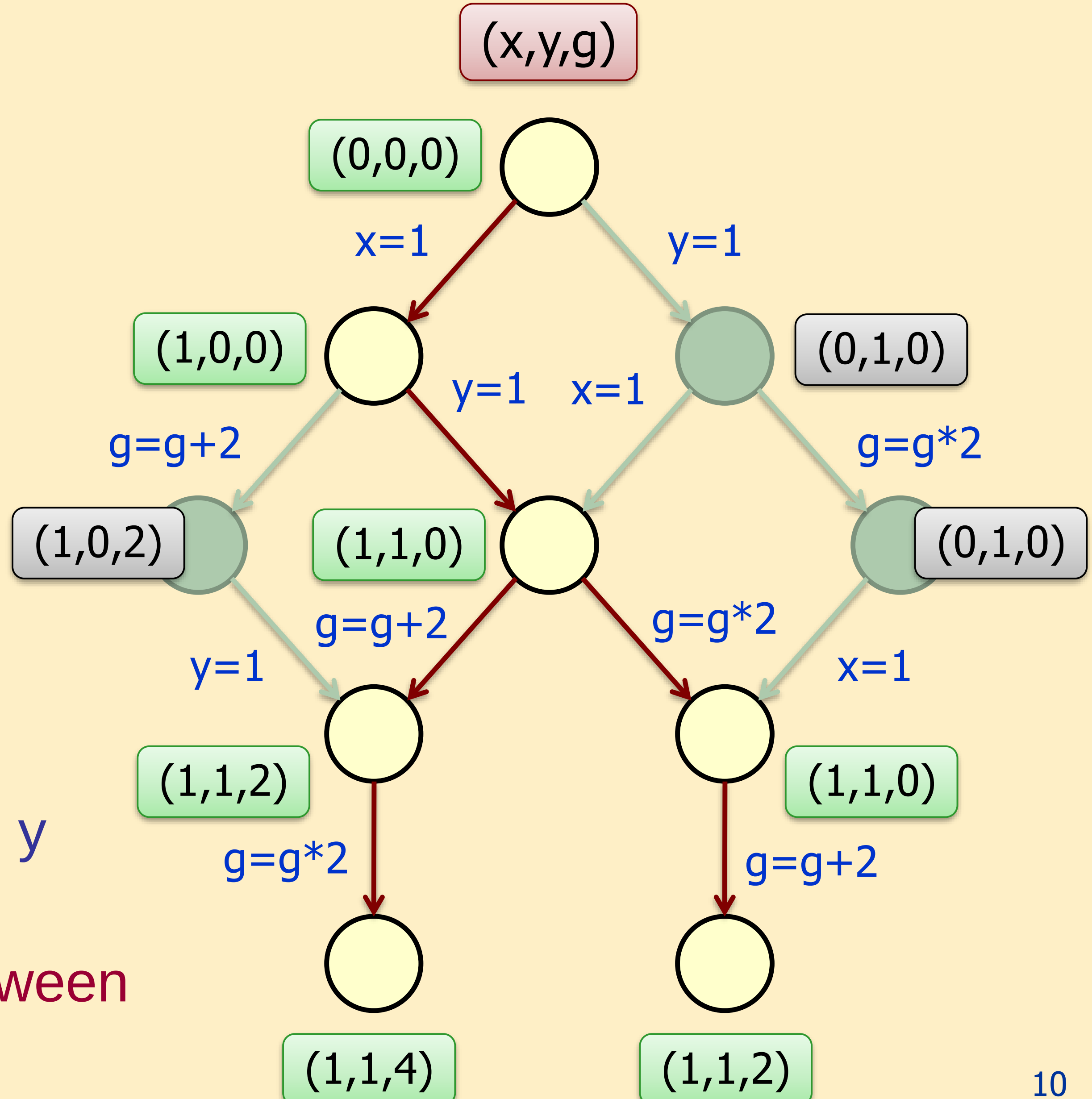
Example: Possible swappings based on data dependency



1. x=1; g=g+2; y=1; g=g*2
2. x=1; y=1; g=g+2; g=g*2
3. y=1; x=1; g=g+2; g=g*2
4. x=1; y=1; g=g*2; g=g+2
5. y=1; x=1; g=g*2; g=g+2
6. y=1; g=g*2; x=1; g=g+2

Example: Representative paths based on data dependency





Local variables: x and y

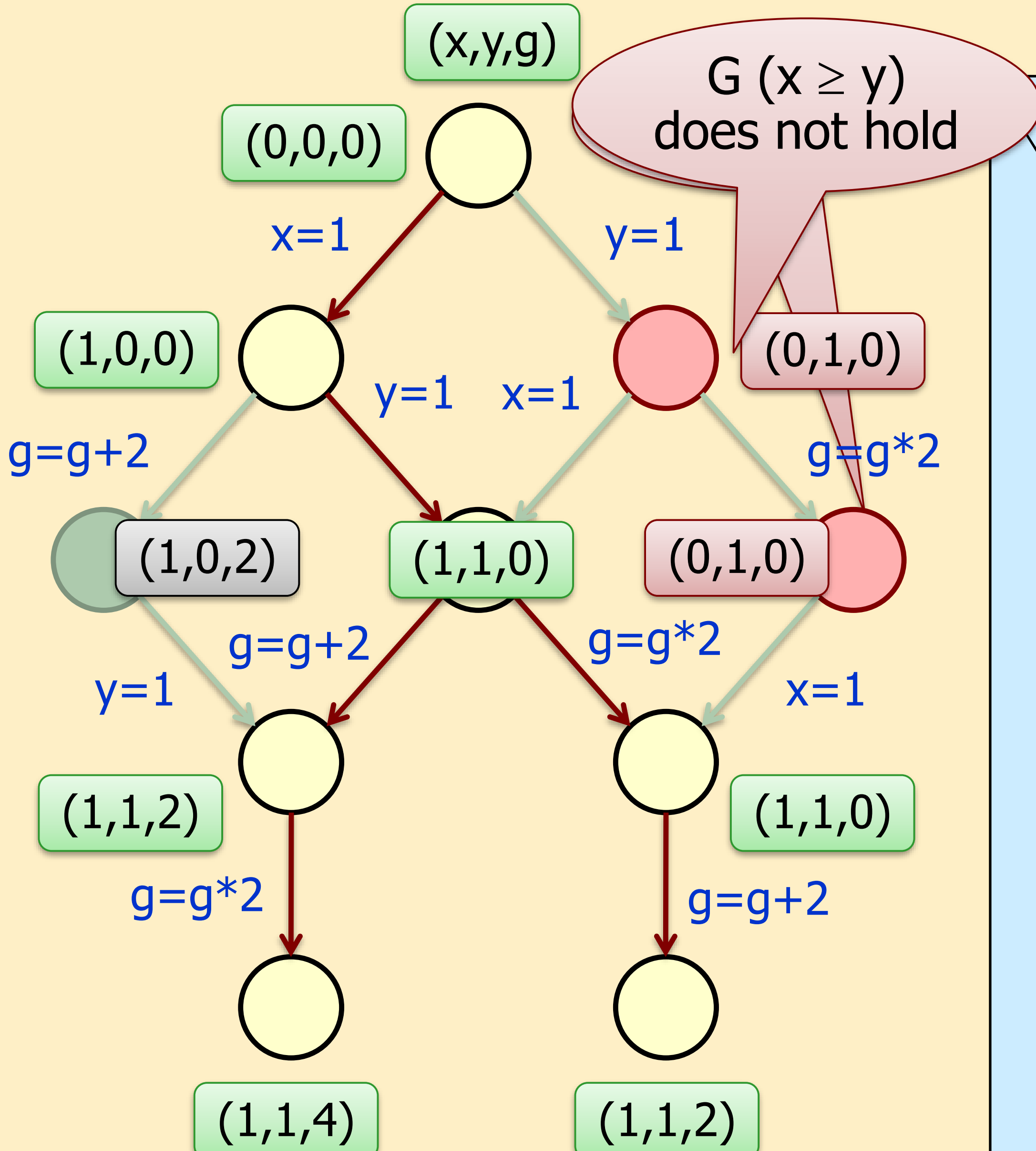
Global variable: g

Data dependency between g=g+2 and g=g*2

Example: Applying partial order reduction

- Reduction
  - "Removing" redundant paths (i.e., only examine remaining, representative paths)
- Reduced graph
  - Remaining paths: Contains possible orderings of non-interchangeable statements due to dependencies
- Correctness of the reduction depends on the goal!
  - Previous reduction: for data dependency
  - Dependency on different property may yield different reduction
    - E.g. $G(x \geq y)$ holds in the previous, reduced graph but not in the original one

Example: G (x≥y) property-based dependency (P)



P: Property dependency

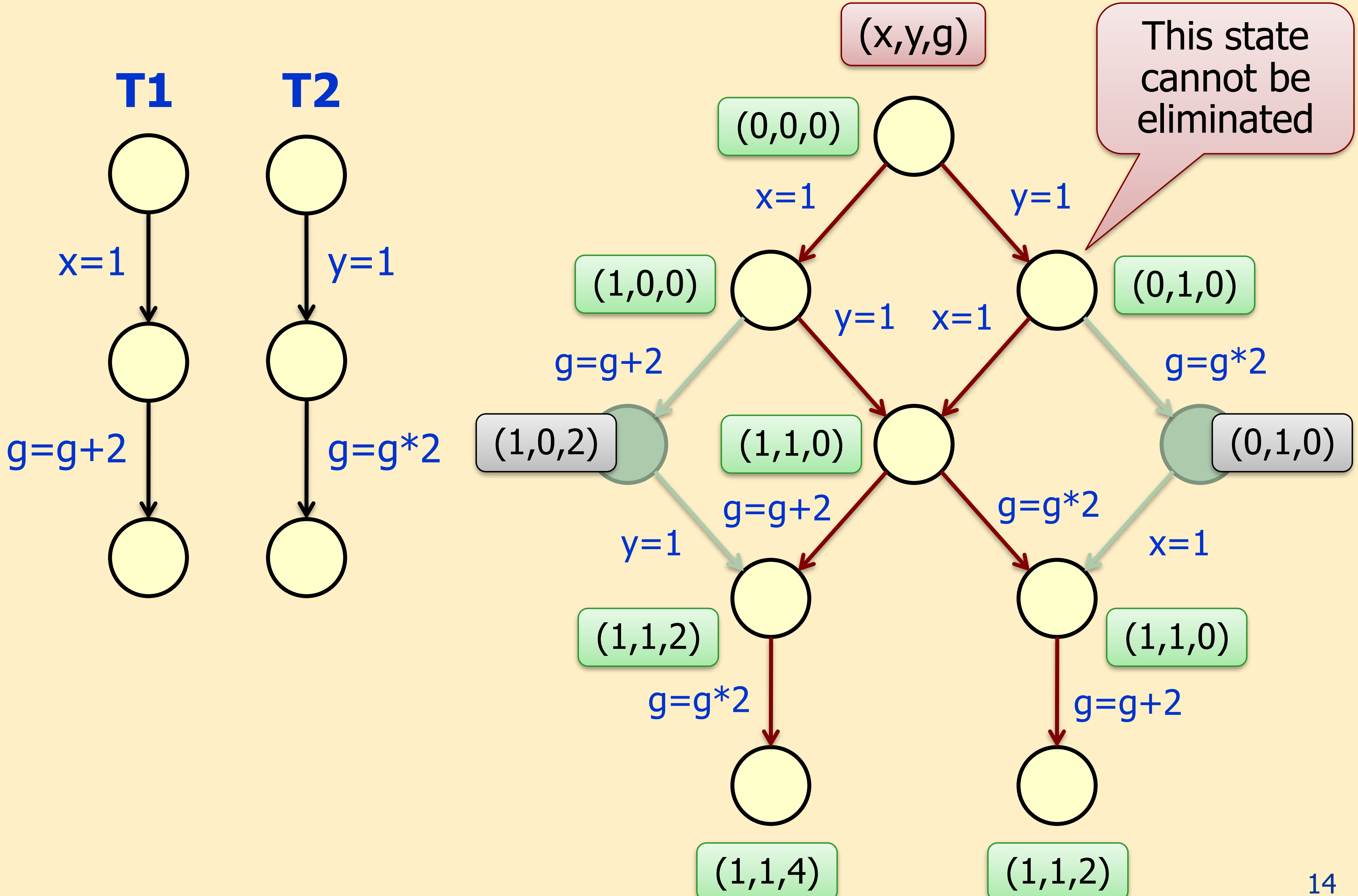
	x=1	y=1	g=g+2	g=g*2
x=1		P	C	I
y=1	P		I	C
g=g+2	C	I		D
g=g*2	I	C	D	

Example: G (x≥y) property preserving reduction

	x=1	y=1	g=g+2	g=g*2
x=1		P	C	I
y=1	P		I	C
g=g+2	C	I		D
g=g*2	I	C	D	

1. x=1; g=g+2; y=1; g=g*2
2. x=1; y=1; g=g+2; g=g*2
3. y=1; x=1; g=g+2; g=g*2
4. x=1; y=1; g=g*2; g=g+2
5. y=1; x=1; g=g*2; g=g+2
6. y=1; g=g*2; x=1; g=g+2

Example: G (x≥y) property preserving reduction



Basis of partial order reduction

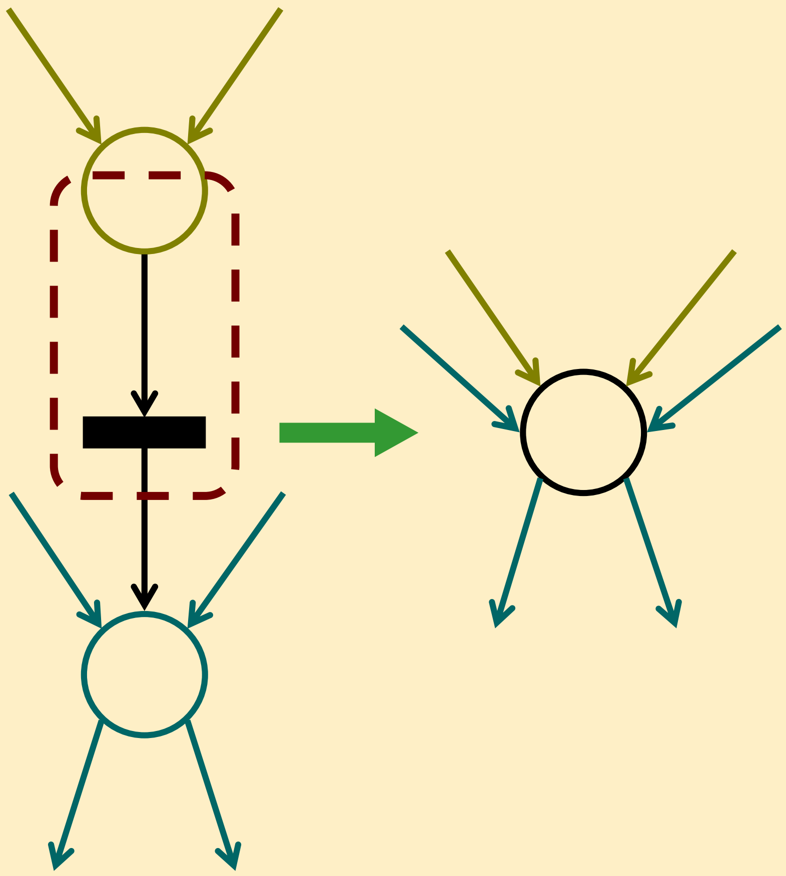
- Two transitions are independent in a state s, if
  - Both are enabled in state s
  - None of their execution disables the other:
    no control dependency (see persistence)
  - The combined effect of the two transitions is
    independent from their order:
    no data or property dependency
- Strong independence
  - Two transitions are strongly independent,
    if they are independent in every state,
    where both are enabled

Structure reduction:
Property preserving model transformations

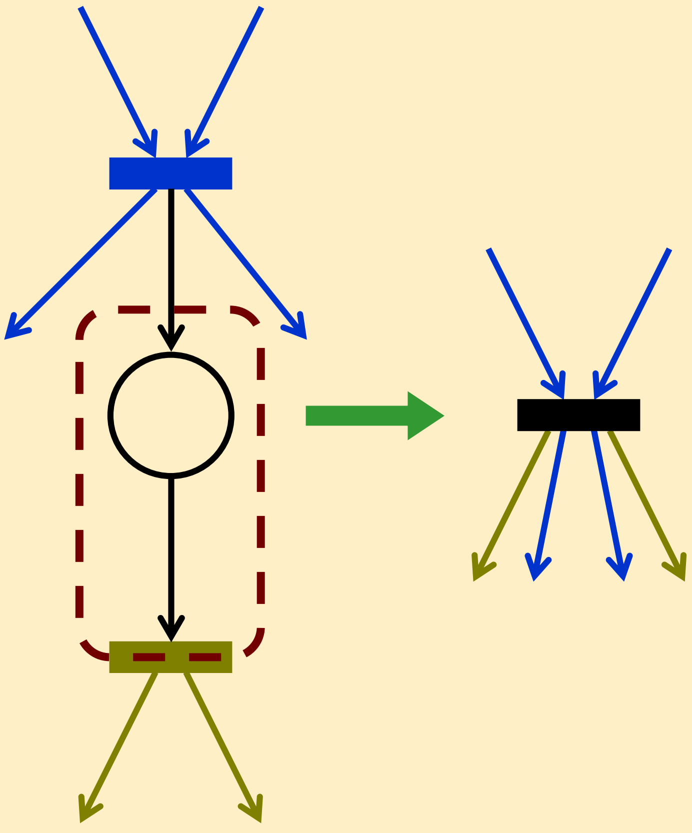
Property preserving transformations

- Structure reduction
  - Goal: reduced model should preserve selected properties
  - A clear model can become compact (hard to understand)
- Simple property preserving transformations:
  - Fusion of series places
  - Fusion of series transitions
  - Fusion of parallel places
  - Fusion of parallel transitions
  - Elimination of self-loop places
  - Elimination of self-loop transitions
- Preserving liveness, boundedness and safeness properties

Rules: Series fusions

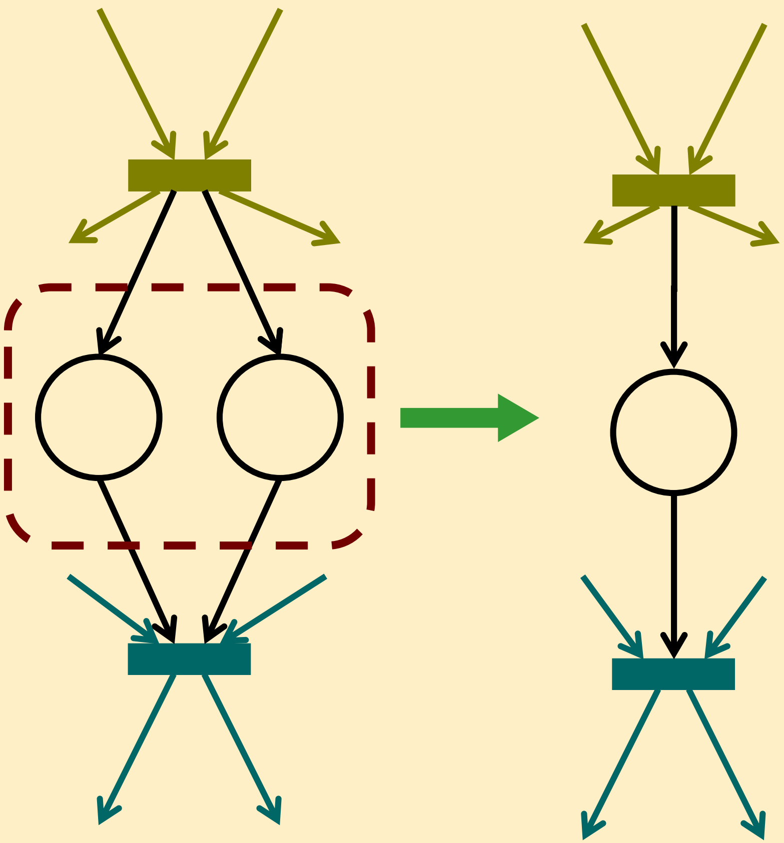


Fusion of series places

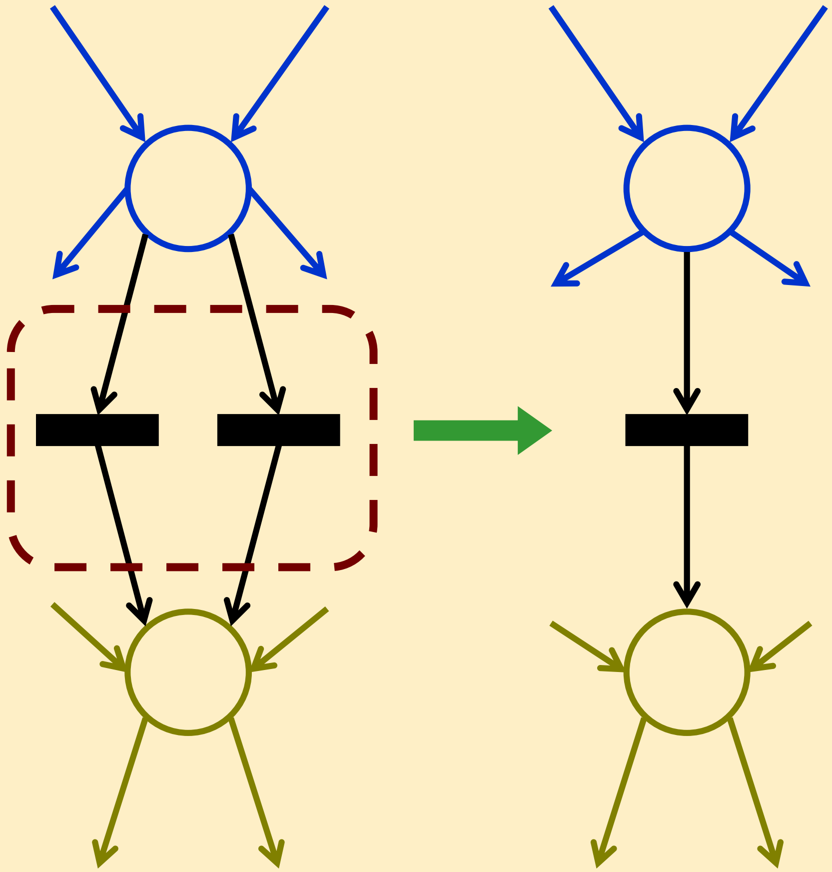


Fusion of series transitions

Rules: Parallel fusions

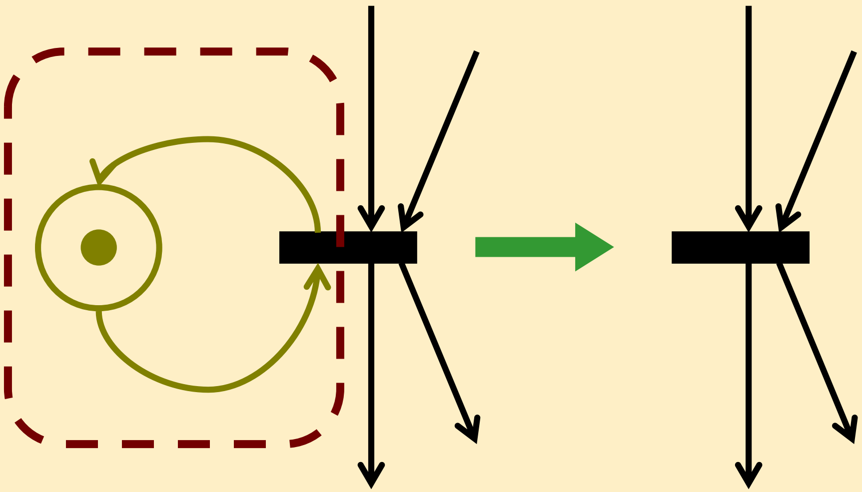


Fusion of parallel places

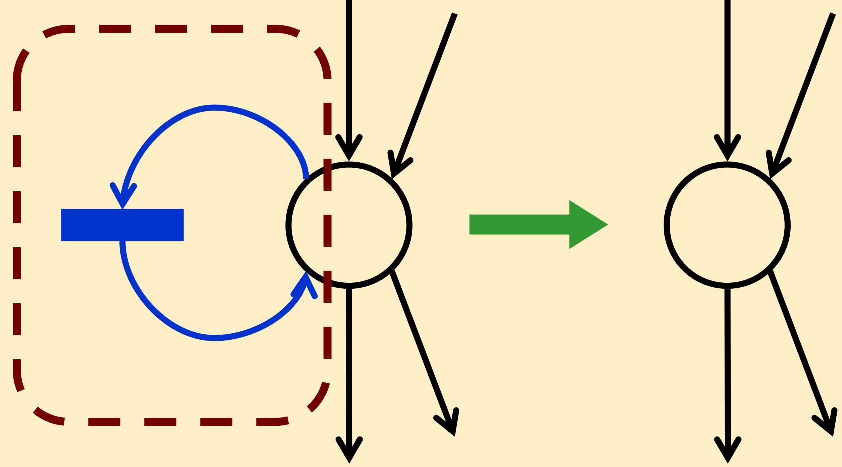


Fusion of parallel transitions

Rules: Elimination of self-loops

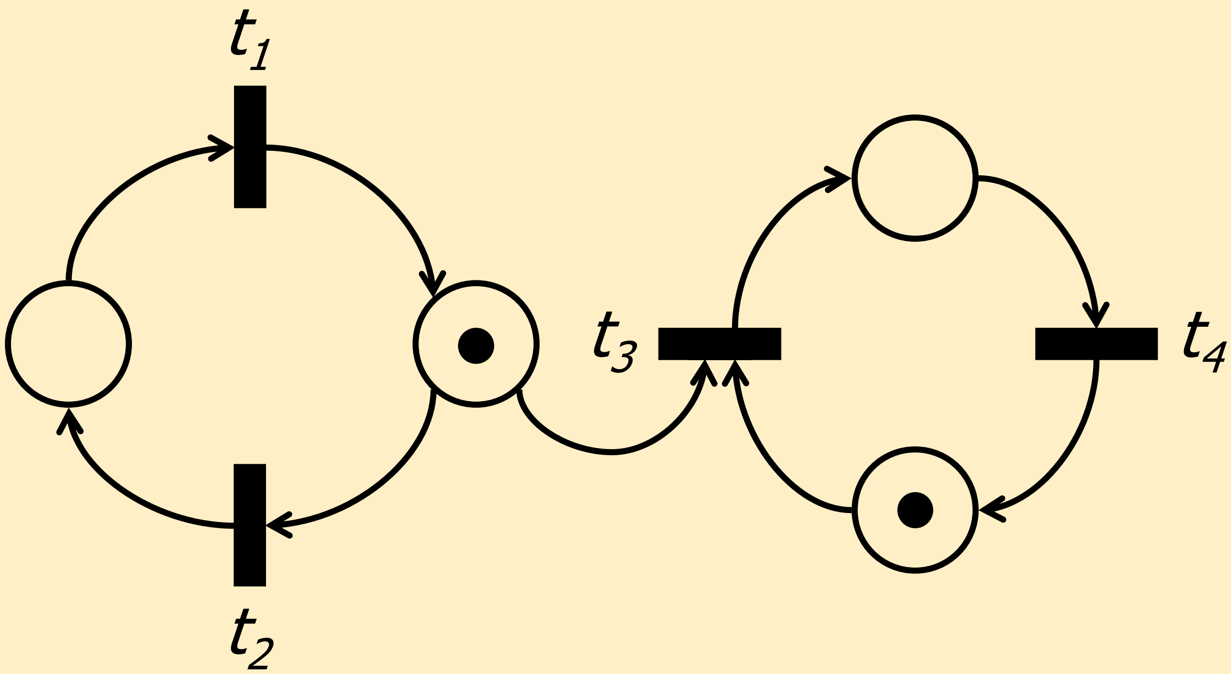


Elimination of self-loop places



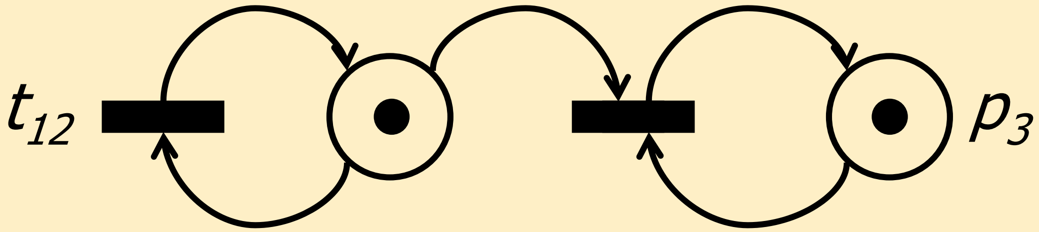
Elimination of self-loop transitions

Example: Step 1



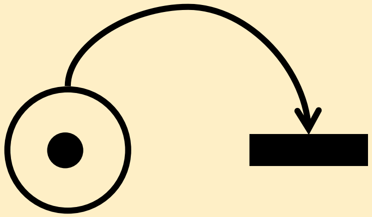
- Fusion of t_2 and t_1 (series transitions) $\rightarrow$ t_{12}
- Fusion of t_3 and t_4 (series transitions) $\rightarrow$ t_{34}

Example: Step 2



- Elimination of t_{12} (self-loop transition)
- Elimination of p_3 (self-loop place)

Example: Result



Bounded, but not live net (and not reversible)