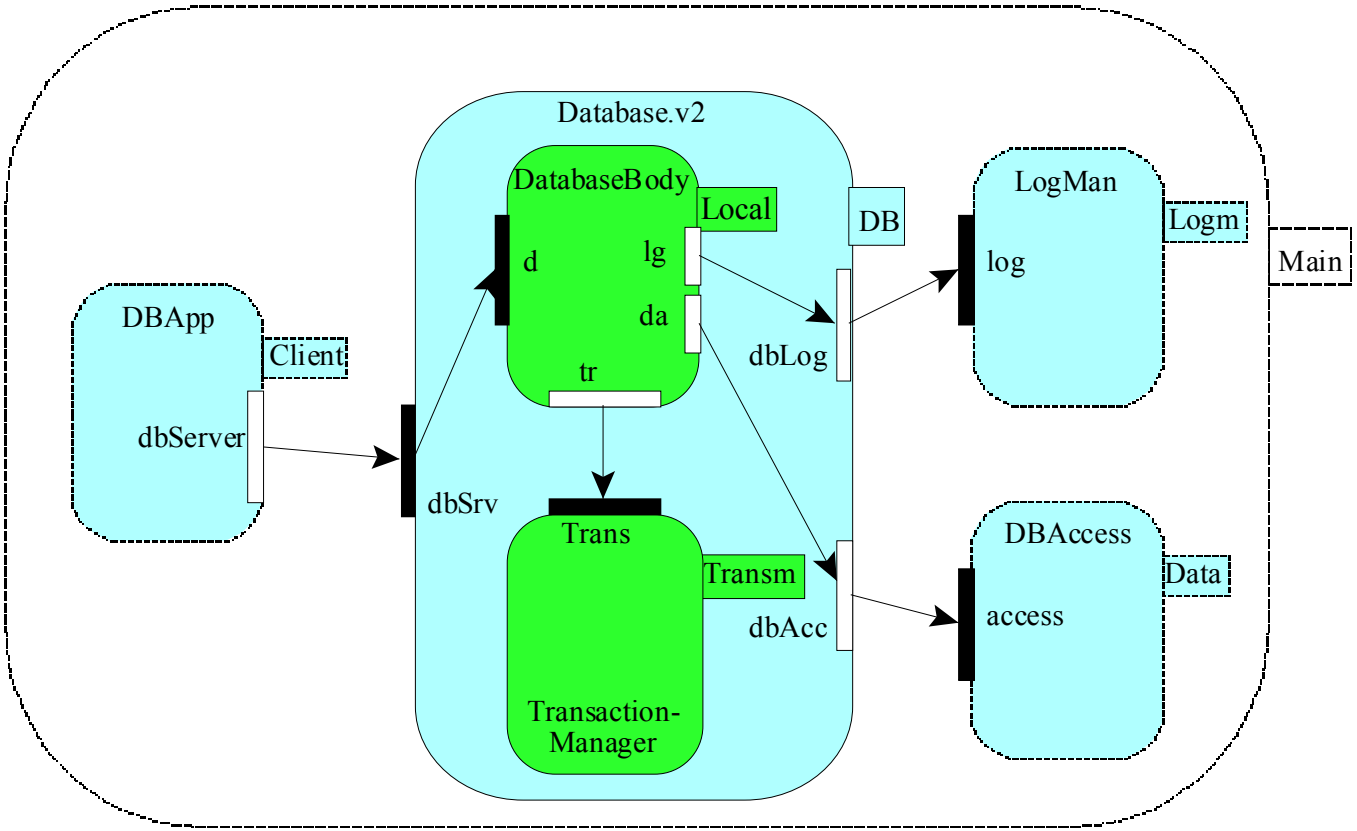
Behavior Protocols Capturing Errors and Updates

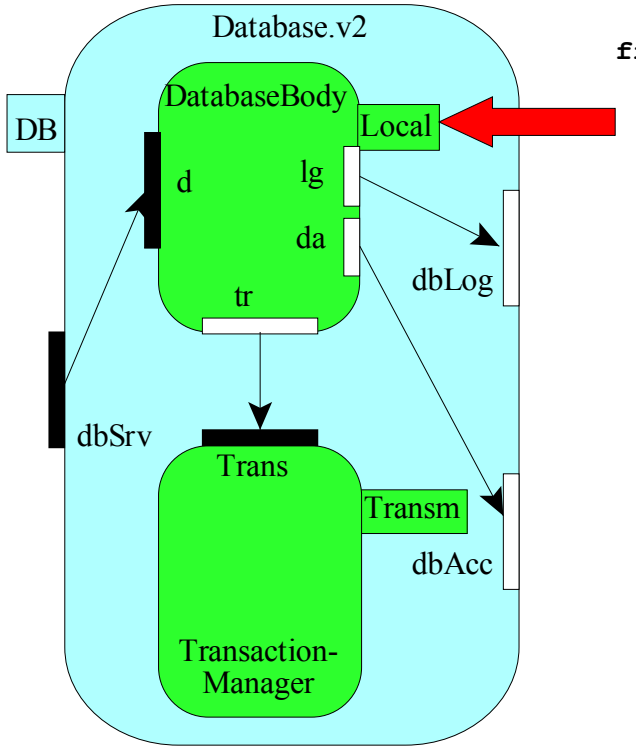
Jiří Adámek, František Plášil

Distributed systems research group
Charles University Prague
http://nenya.ms.mff.cuni.cz

SOFA project



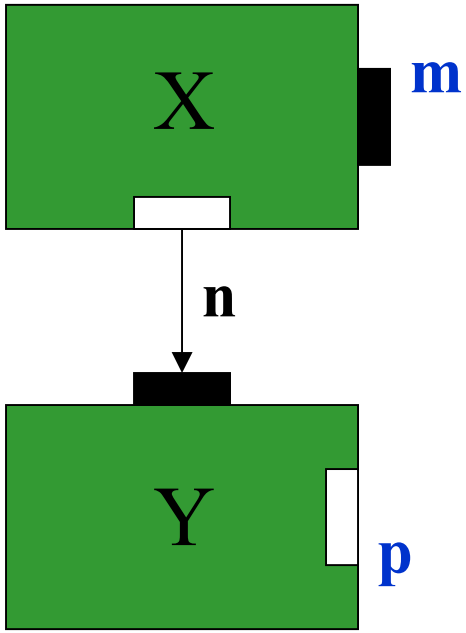
SOFA behavior protocols




```
frame protocol:
       !da.Open ;
       (   ?d.Insert { !tr.Begin ; !da.Insert ;
              !lg.LogEvent; (!tr.Commit +
           !tr.Abort) }
           +
           ?d.Delete { !tr.Begin ; !da.Delete ;
              !lg.LogEvent;
              (!tr.Commit + !tr.Abort) }
           +
           ?d.Query { !da.Query }
       )* ;
       !da.Close
```

Plasil,F. Visnovsky: Behavior protocols for SW components.
IEEE Trans. on SE, Nov. 2002

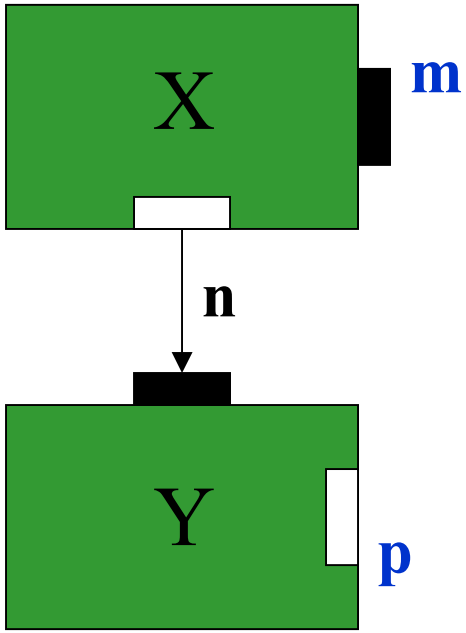
Behavior protocols



$\mathbf{Prot_X = (?m\uparrow ; ((!n\uparrow ; ?n\downarrow) + NULL) ; !m\downarrow)^*}$

$\mathbf{Prot_Y = (?n\uparrow ; !p\uparrow ; ?p\downarrow ; !n\downarrow)^*}$

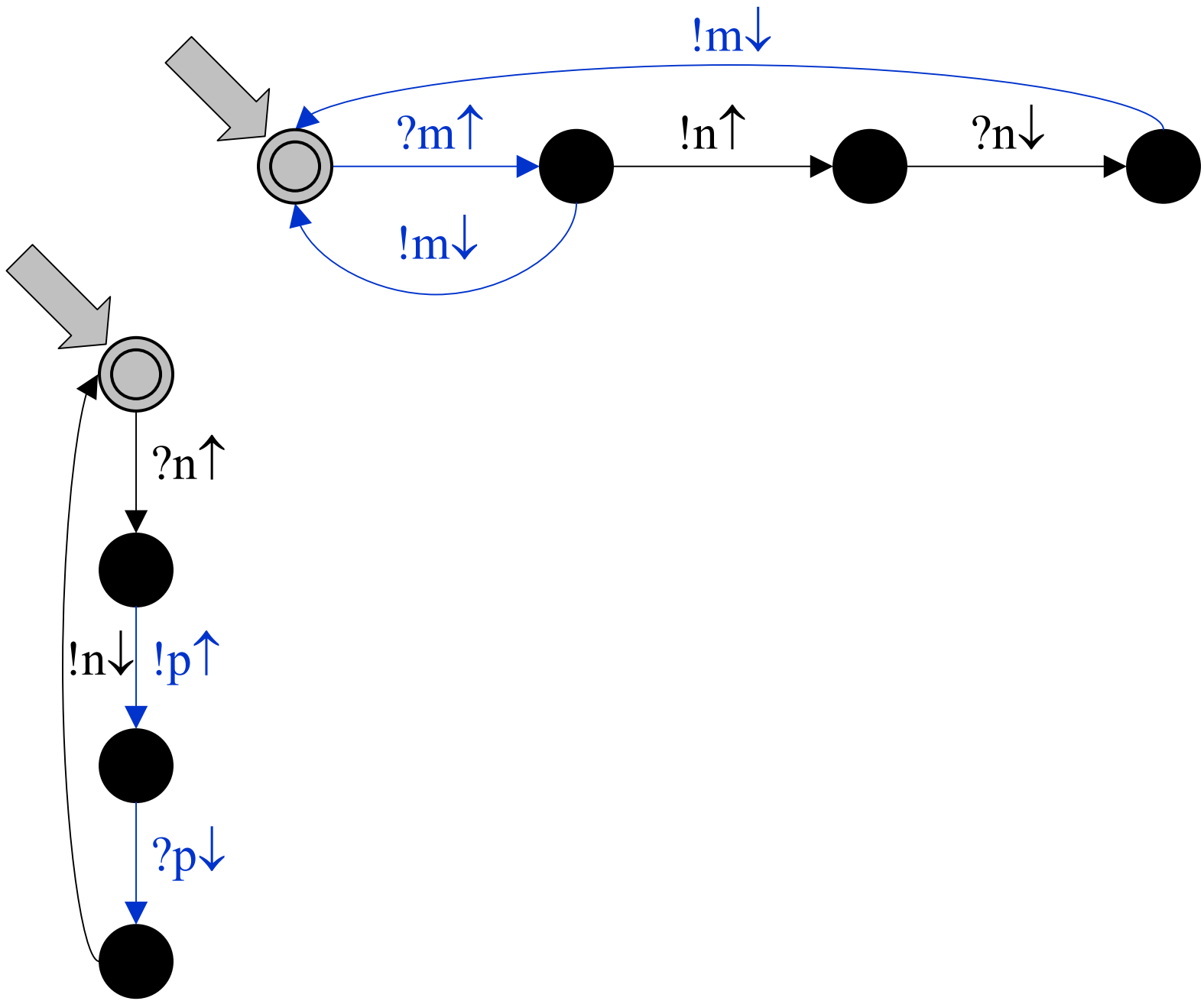
Behavior protocols

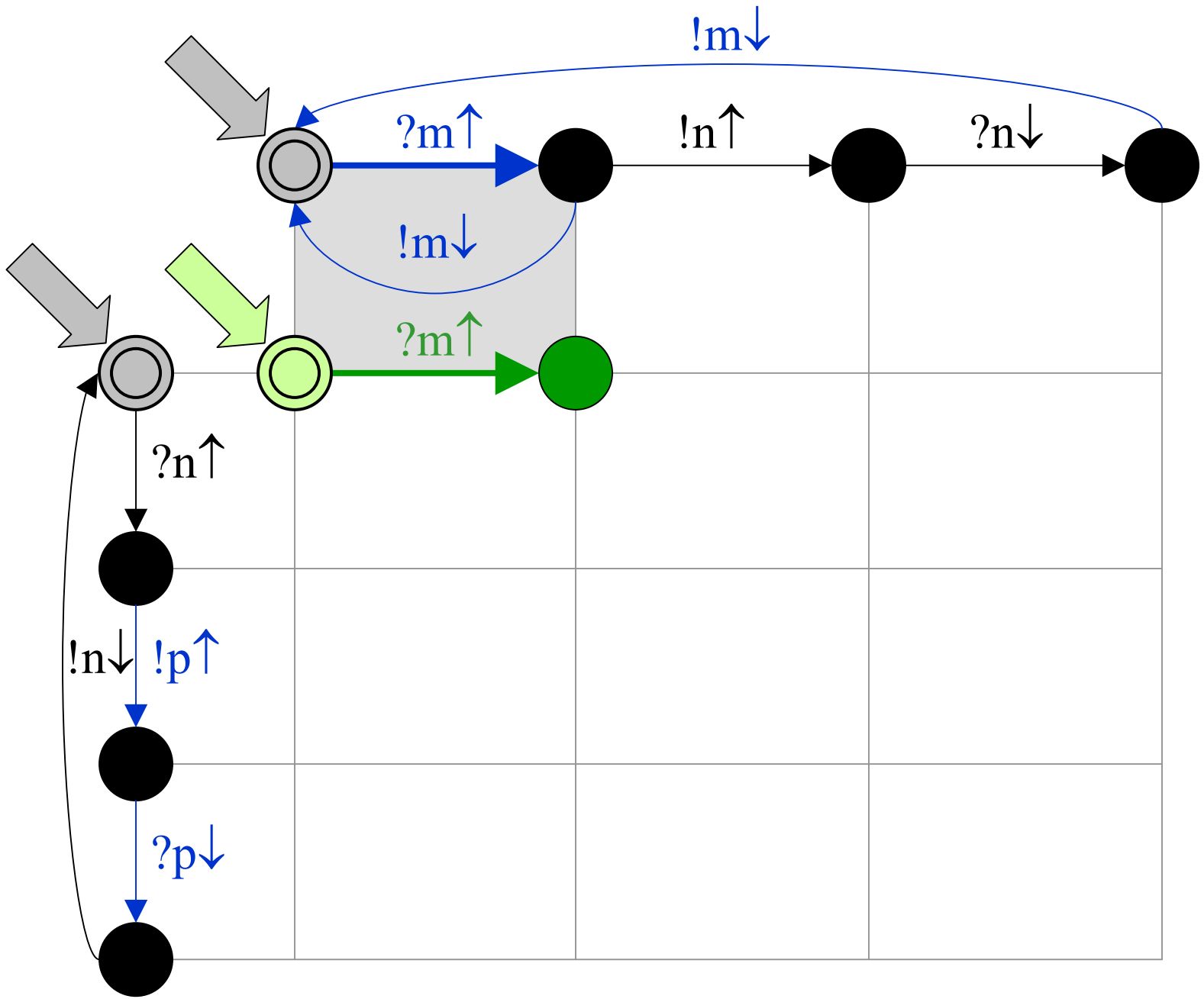


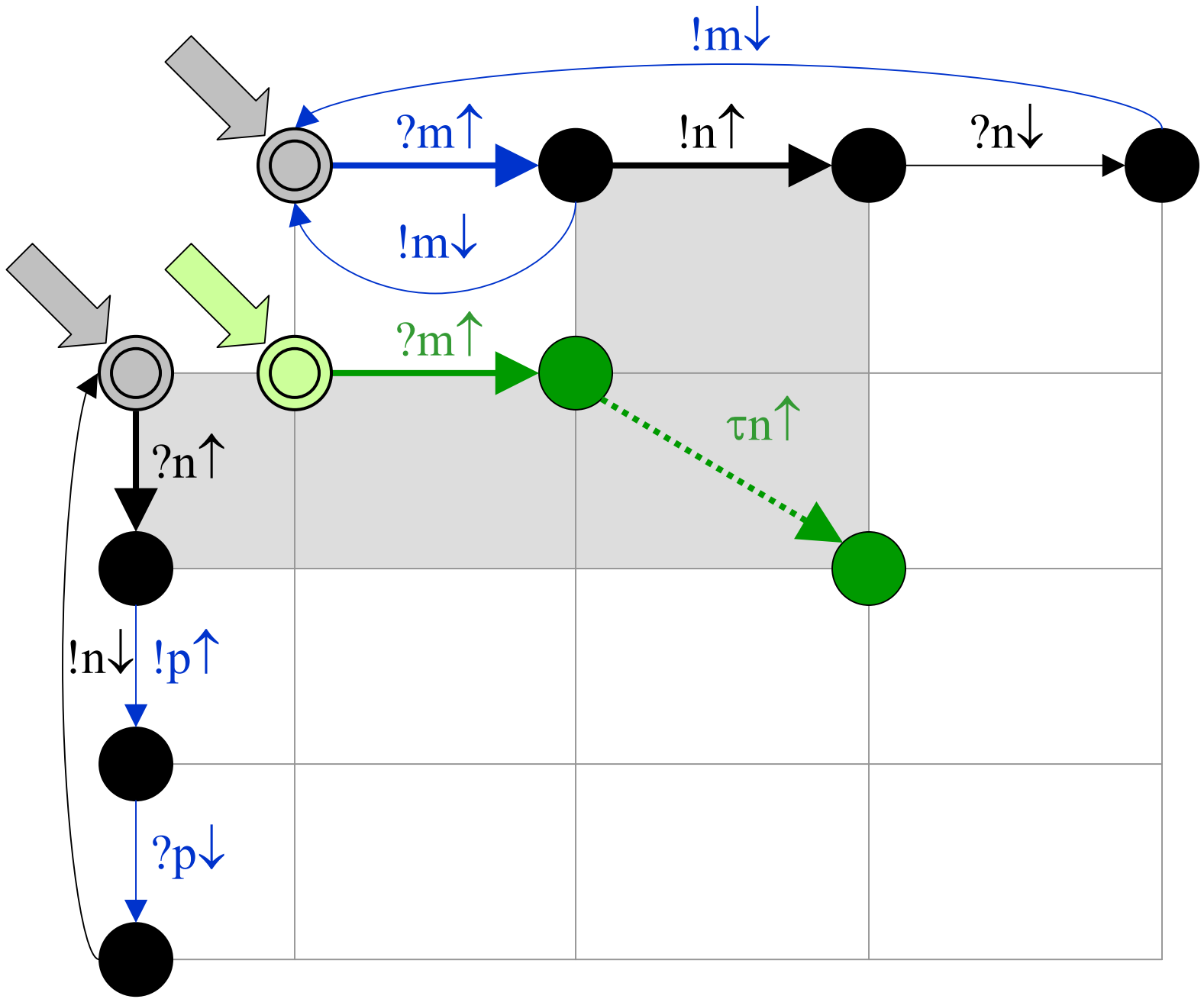
$Prot_X$ = (?m↑ ; ((!n↑ ; ?n↓) + NULL) ; !m↓)*

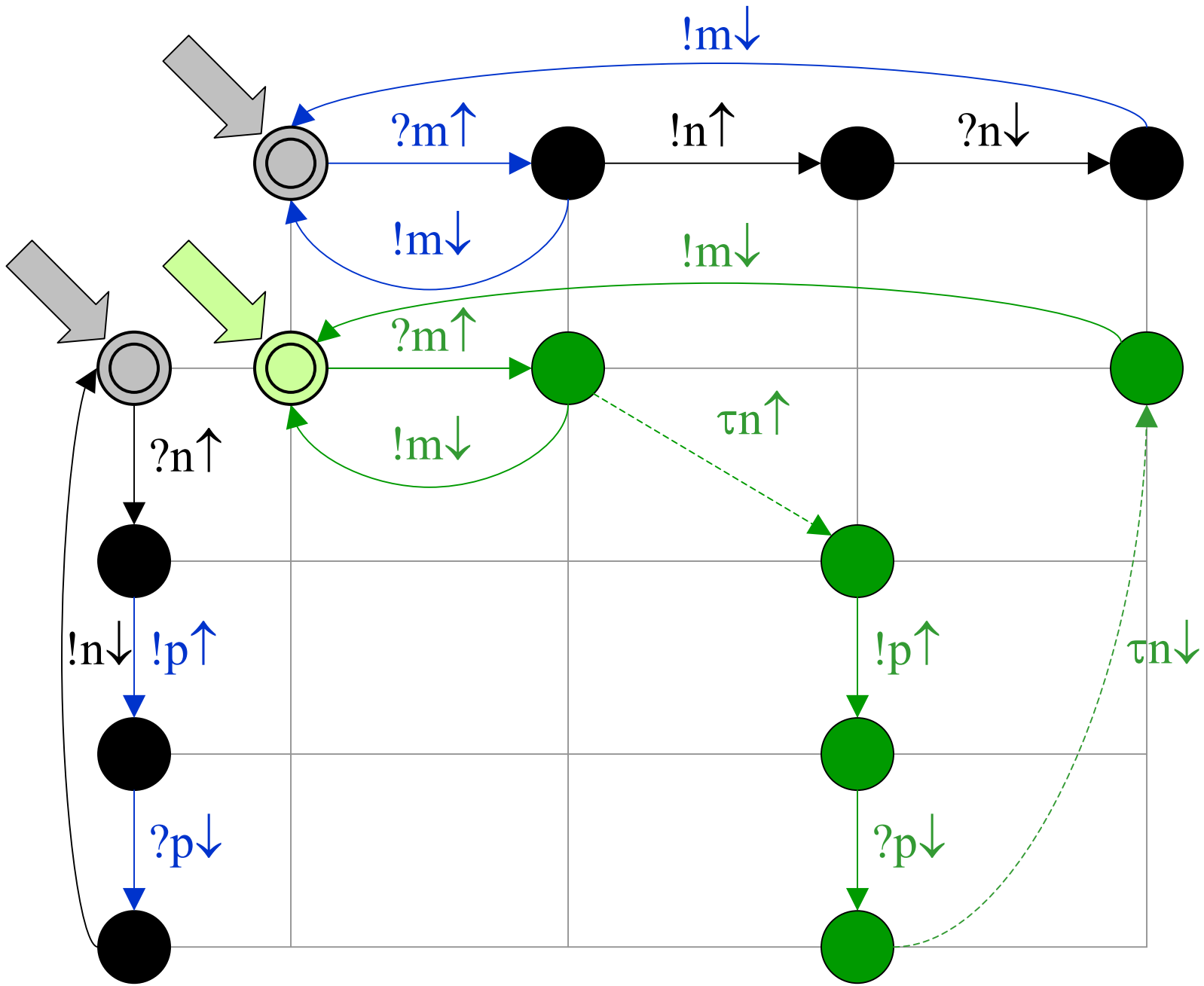
$Prot_Y$ = (?n↑ ; !p↑ ; ?p↓ ; !n↓)*

$Prot_X \nabla_{\{n\uparrow, n\downarrow\}} Prot_Y$ = { < ?m↑ ; τn↑ ; !p↑ ; ?p↓ ; τn↓ ; !m↓ >,
< ?m↑ ; !m↓ >, … }


!m↓
?m↑
!n↑
?n↓
!m↓
?n↑
!n↓
!p↑
?p↓


!m↓
?m↑
!n↑
?n↓
!m↓
?m↑
?n↑
!n↓
!p↑
?p↓

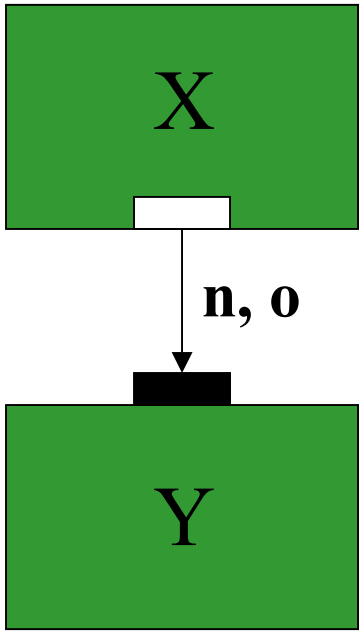

!m↓
?m↑
!n↑
?n↓
!m↓
?m↑
τn↑
?n↑
!n↓
!p↑
?p↓


!m↓
?m↑
!n↑
?n↓
!m↓
!m↓
?m↑
τn↑
?n↑
!m↓
!n↓
!p↑
!p↑
τn↓
?p↓
?p↓

Composition errors

- Bad activity - $\varepsilon n\uparrow$, $\varepsilon n\downarrow$
- No activity (deadlock) - $\varepsilon\varnothing$
- Infinite activity (divergence) - $\varepsilon\propto$

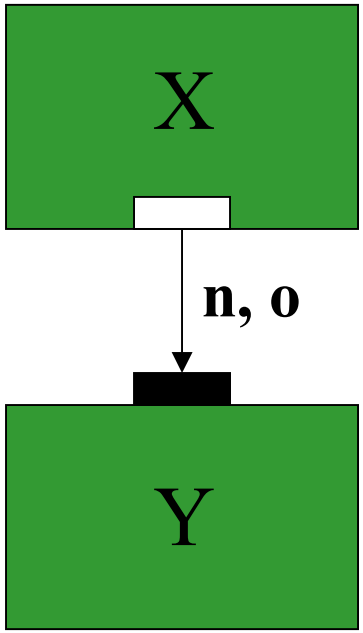
Bad activity



$\mathbf{Prot_X = !n\uparrow\ ;\ ?n\downarrow\ ;\ ((\ !o\uparrow\ ;\ ?o\downarrow\ )\ +\ (\ !n\uparrow\ ;\ ?n\downarrow\ ))}$

$\mathbf{Prot_Y = ?n\uparrow\ ;\ !n\downarrow\ ;\ ?o\uparrow\ ;\ !o\downarrow}$

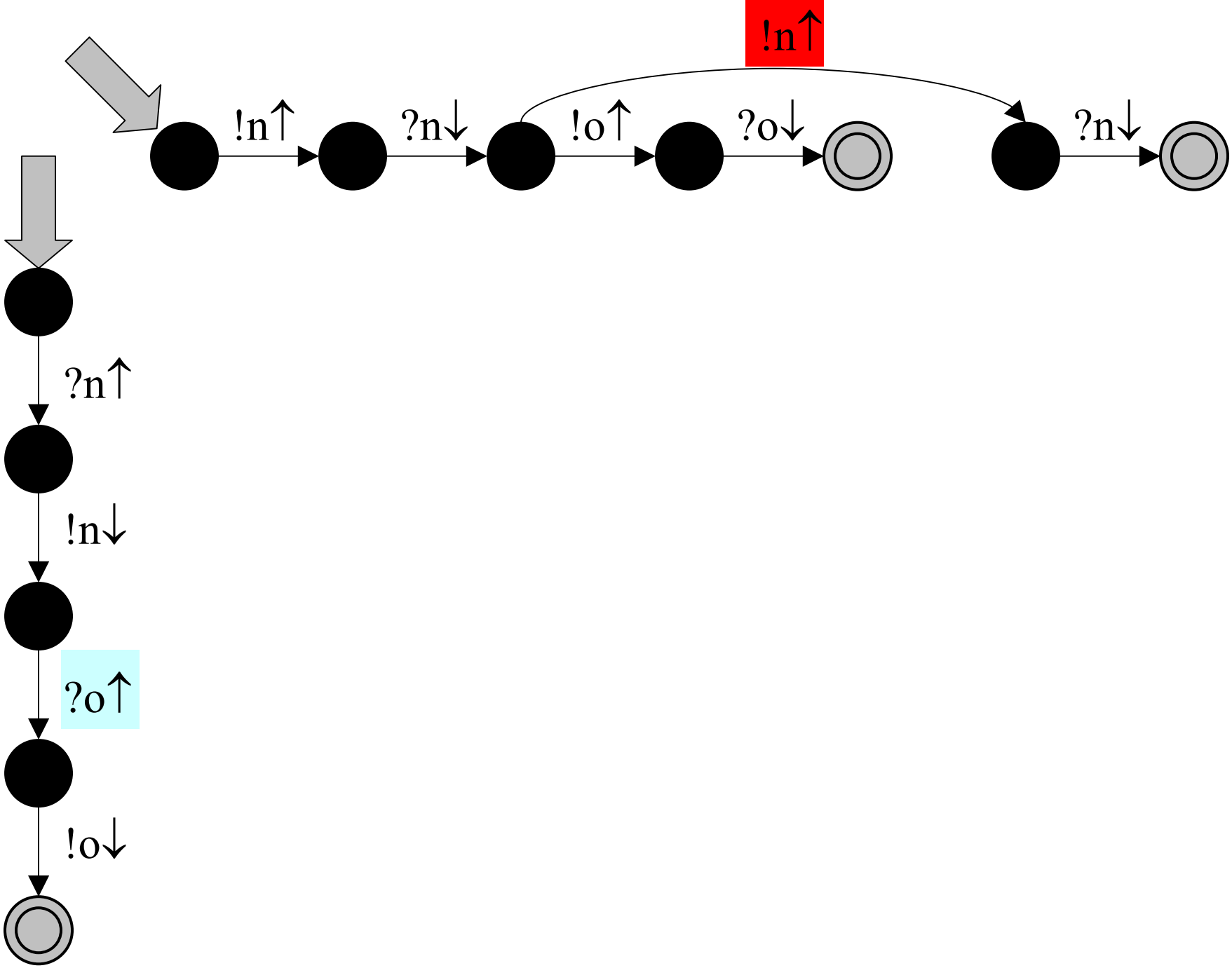
Bad activity

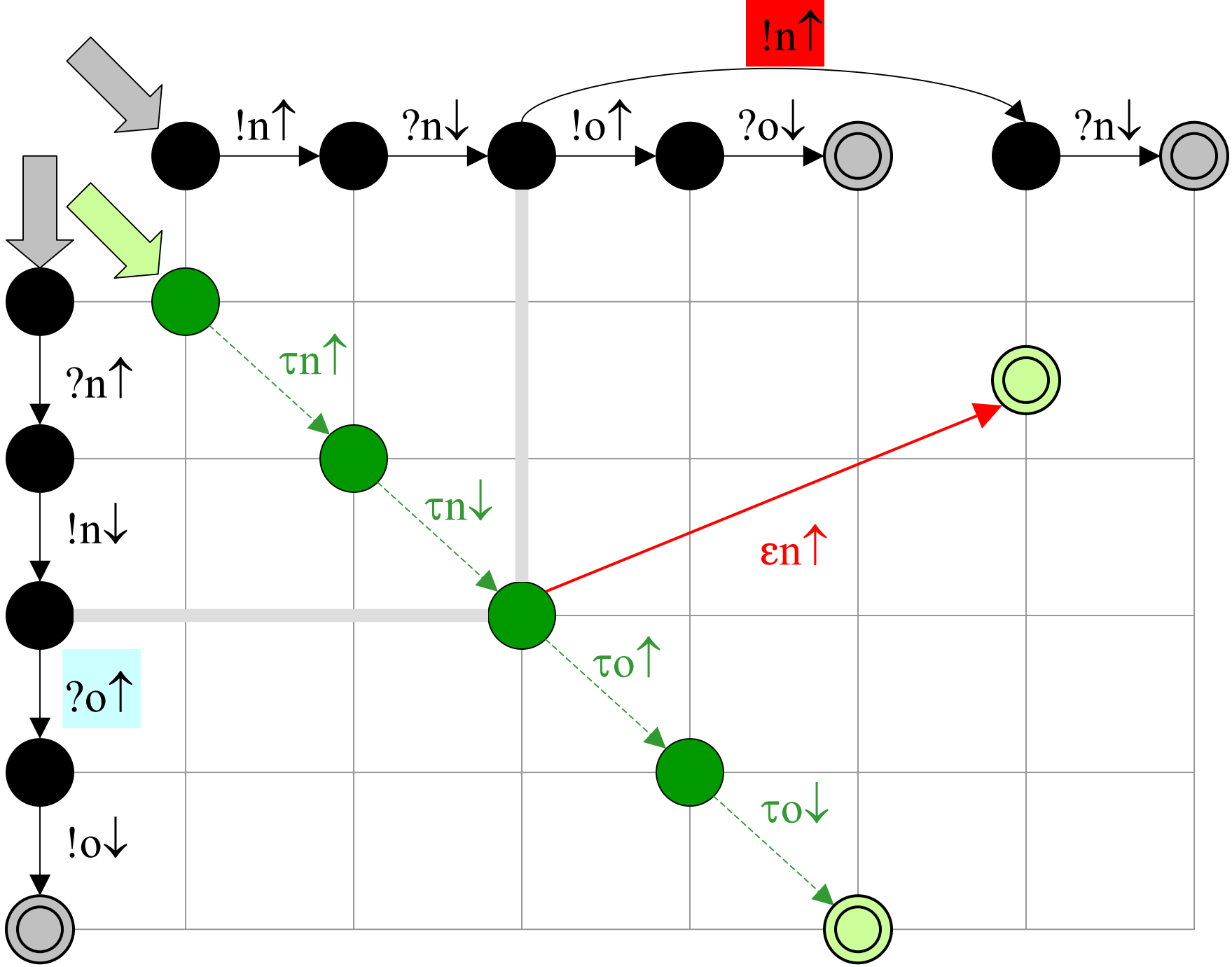


$\mathbf{Prot_X = !n\uparrow\ ;\ ?n\downarrow\ ;\ (\ (\ !o\uparrow\ ;\ ?o\downarrow\ )\ +\ (\ !n\uparrow\ ;\ ?n\downarrow\ )\ )}$

$\mathbf{Prot_Y = ?n\uparrow\ ;\ !n\downarrow\ ;\ ?o\uparrow\ ;\ !o\downarrow}$

$$\mathbf{Prot_X \nabla_{\{\, n\uparrow,\, n\downarrow,\, !o\uparrow,\, o\downarrow \,\}} Prot_Y = \{\ <\ \tau n\uparrow\ ;\ \tau n\downarrow\ ;\ \tau o\uparrow\ ;\ \tau o\downarrow\ >,\ <\ \tau n\uparrow\ ;\ \tau n\downarrow\ ;\ \varepsilon n\uparrow\ >\ \}}$$

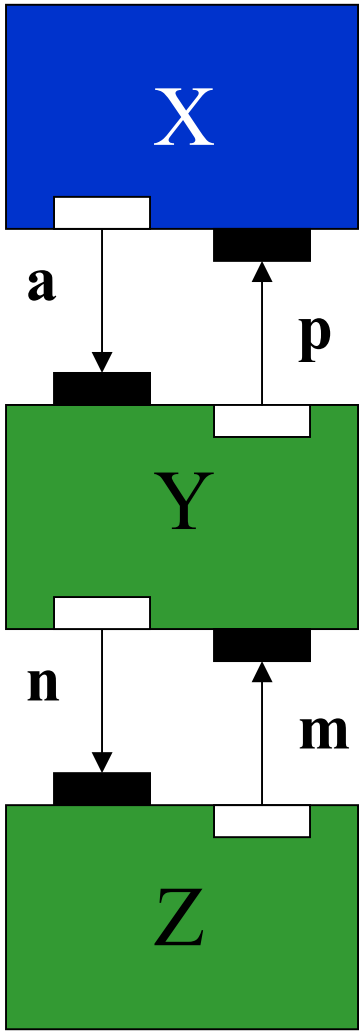

!n↑
!n↑
?n↓
!o↑
?o↓
?n↓
?n↑
!n↓
?o↑
!o↓


!n↑
!n↑
?n↓
!o↑
?o↓
?n↓
?n↑
τn↑
τn↓
εn↑
!n↓
?o↑
τo↑
τo↓
!o↓

Consent operator ∇

- Constructs behavior protocols
  - of composed components
- Captures
  - composition errors
- Checks
  - atomicity of dynamic updates

Dynamic update (1)

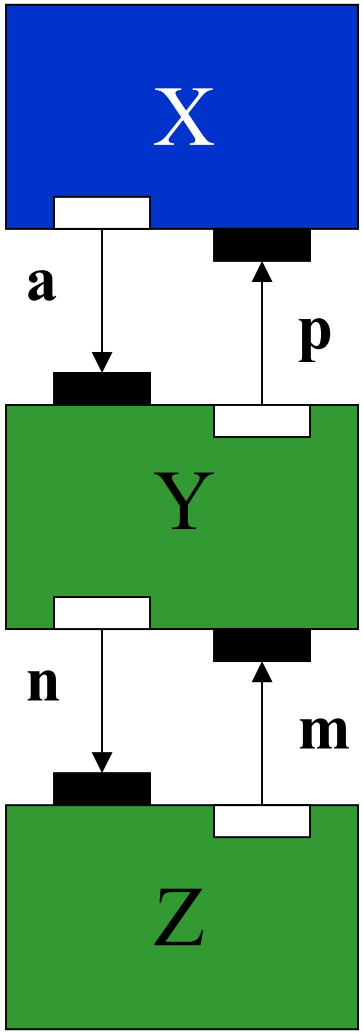


$\mathbf{Prot_X = ((!a\ ;\ ?p)^*\ ;\ ?\pi_1\uparrow\ ;\ !\pi_1\downarrow)^*}$

$\mathbf{Prot_Y = (?a\ ;\ !p)^*\ |\ (?m\ ;\ !n)^*}$

$\mathbf{Prot_Z = (!m\ ;\ ?n)^*}$

Dynamic update (1)



$\mathbf{Prot_X = ((!a ; ?p)^* ; ?\pi_1\uparrow ; !\pi_1\downarrow)^*}$

$\mathbf{Prot_Y = (?a ; !p)^* \mid (?m ; !n)^*}$

$\mathbf{Prot_Z = (!m ; ?n)^*}$

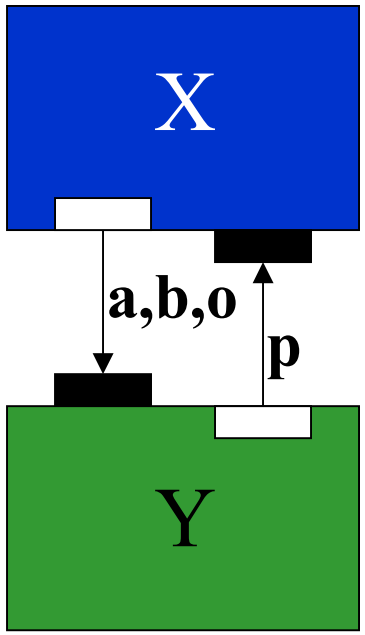
⇩ $\mathbf{(Prot_X \nabla_{\{a\uparrow, a\downarrow, p\uparrow, p\downarrow\}} Prot_Y) \nabla_{\{n\uparrow, n\downarrow, m\uparrow, m\downarrow\}} Prot_Z}$

$\mathbf{((\tau a ; \tau p)^* ; \tau\pi_1\uparrow ; \tau\pi_1\downarrow)^* \mid (\tau m ; \tau n)^*}$

Dynamic update (2)

$\mathbf{Prot_X = ((\ !a\ ;\ ?\pi_1\uparrow\ ;\ !\pi_1\downarrow\ )\ +\ (\ !b\ ;\ ?\pi_2\uparrow\ ;\ !\pi_2\downarrow\ ))\ ;\ (\ !o\ |\ ?p\ )}$

$\mathbf{Prot_Y = (\ ?a\ ;\ ?o\ ;\ !p\ )\ +\ (\ ?b\ ;\ (\ ?o\ |\ !p))}$

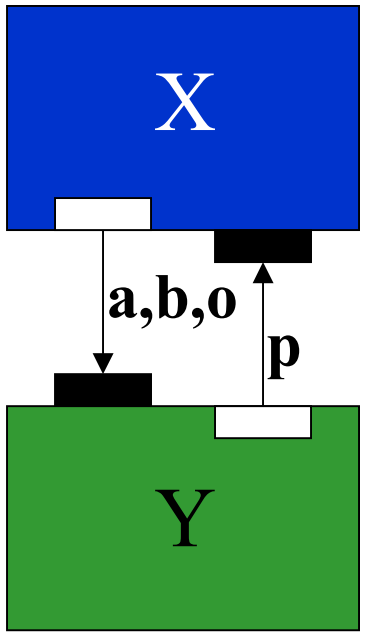


Dynamic update (2)

$\mathbf{Prot_X = ((\ !a\ ;\ ?\pi_1\uparrow\ ;\ !\pi_1\downarrow\ ) + (\ !b\ ;\ ?\pi_2\uparrow\ ;\ !\pi_2\downarrow\ ))\ ;\ (\ !o \mid ?p\ )}$

$\mathbf{Prot_Y = (\ ?a\ ;\ ?o\ ;\ !p\ ) + (\ ?b\ ;\ (\ ?o \mid !p))}$

⇩ $\mathbf{Prot_X \nabla_{\{\ldots\}} Prot_Y}$



$$\mathbf{(\ \tau a\ ;\ \tau\pi_1\uparrow\ ;\ \tau\pi_1\downarrow\ ;\ \tau o\ ;\ \tau p\ ) +}$$
$$\mathbf{(\ \tau b\ ;\ (}$$
$$\mathbf{(\ \tau\pi_2\uparrow\ ;\ \tau\pi_2\downarrow\ ;\ (\tau o \mid \tau p\ )) + (\ \tau\pi_2\uparrow\ ;\ \varepsilon p\uparrow\ )}$$
$$\mathbf{)}$$
$$\mathbf{)}$$

Conclusion

- Atomicity of component updates
  - tested statically
  - violation indicated by ∇
- No locking
  - ~performance benefit
  - deadlock avoidance
- Key benefit: Info on updates
  - in protocol
  - outside of code

Discussion

- Simplifies code
  - Coordination of locking on multiple interfaces skipped
- Dynamic checker used anyway
  - Let it check whether update comes at the right time
- Nested components
  - Compliance def. modification (technical, not principle)
    - Strictly synchronized with children's willingness to be updated
    - Wait until children finish internal actions (~liveness property)
- Internal threads
  - Assumption: Implementation of X respects updating plan in $Prot_X$