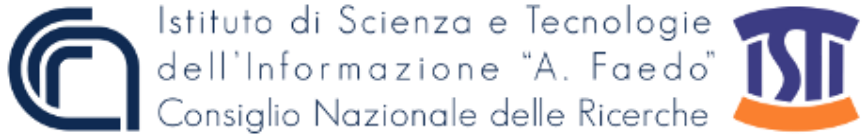
Research Challenges in Orchestration Synthesis

Davide Basile, Maurice H. ter Beek

(ICE 2023)



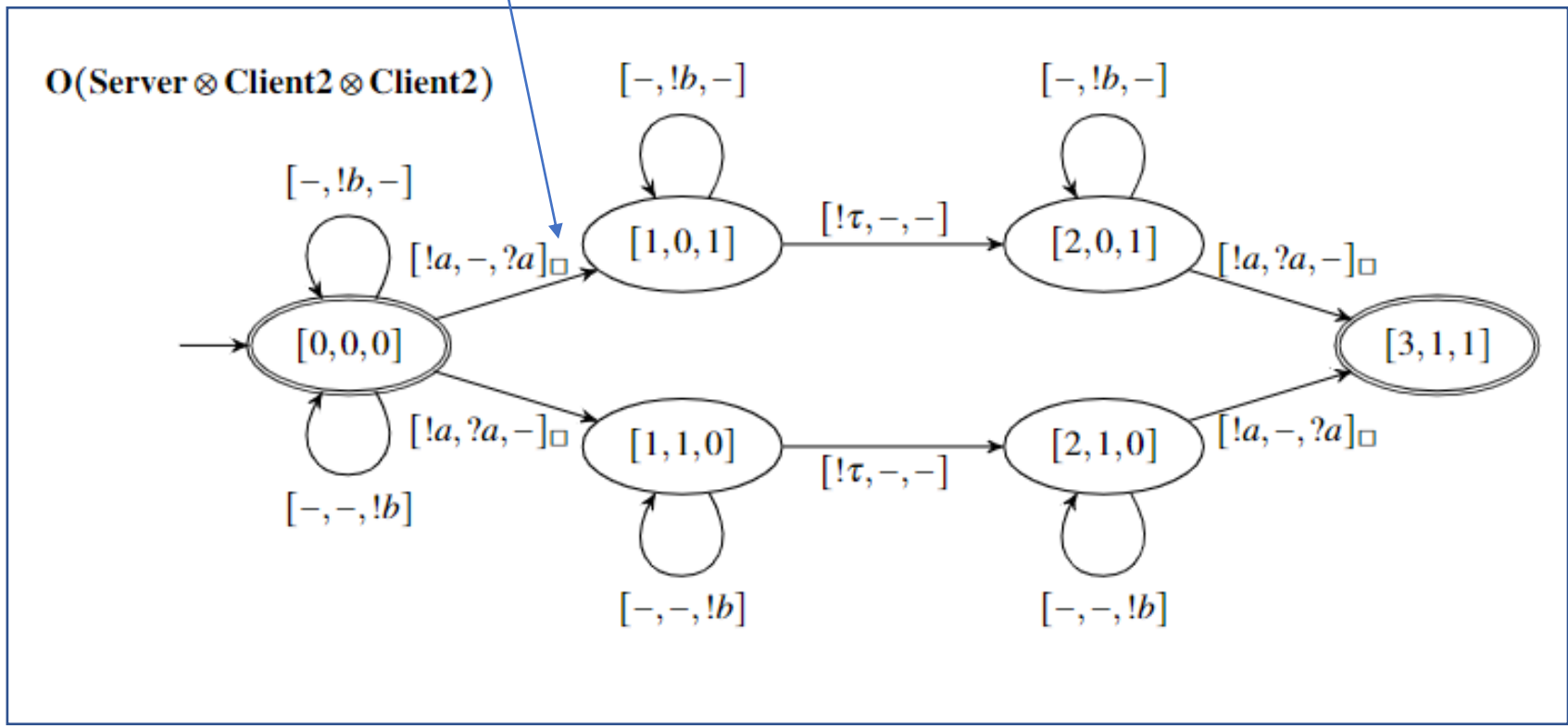
Overview

- Introduction to contract automata
- Orchestration synthesis algorithm
- Research Challenges
- Conclusion

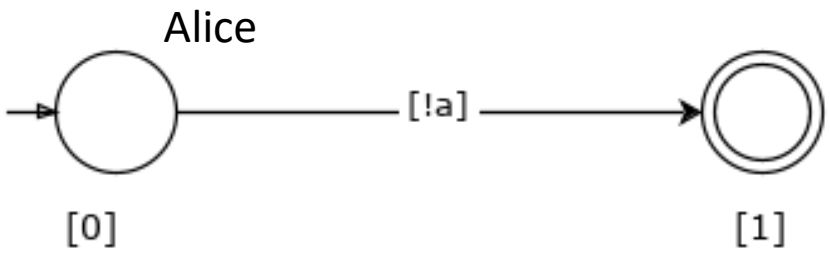
Contract Automata

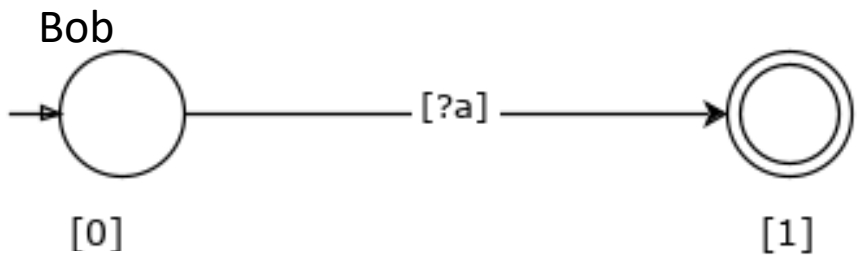
- Behavioural contracts: used to specify the interactions of services
- Contract automata: a dialect of finite-state automata modelling behavioural contracts
- Introduced almost 10 years ago
  - Basile, D., Degano, P. and Ferrari, G.L. Automata for analysing service contracts. In *TGC 2014.*
  - Basile, D., Degano, P., Ferrari, G.L. and Tuosto, E. From orchestration to choreography through contract automata. In *ICE 2014*.
- We will present challenges in the orchestration synthesis of contract automata

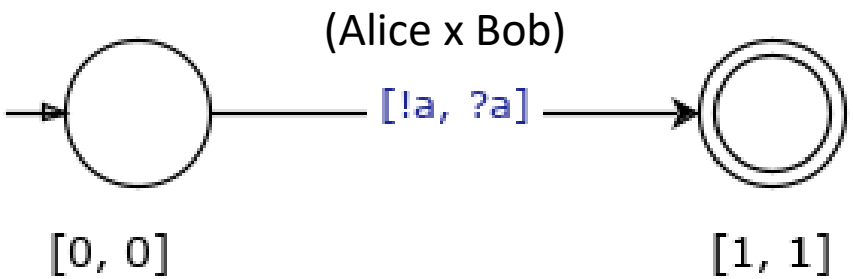
- Contract automata are FSA enhanced with:
  - Partitioned alphabet of actions:
    - offers !a (A^o) and requests ?a (A^r)
    - special idle action (- not in $A^o \cup A^r$)
  - rank : the number of services in the contract
  - Transitions partitioned into *optional* and *necessary*
  - States are list of basic states
  - Labels are list of actions and are constrained to be:
    - offers: (-, -, -, !a)
    - requests: (-, ?a, -, -)
    - matches: (-, ?a, -, !a)
      (only between two)
  - size(list) = rank

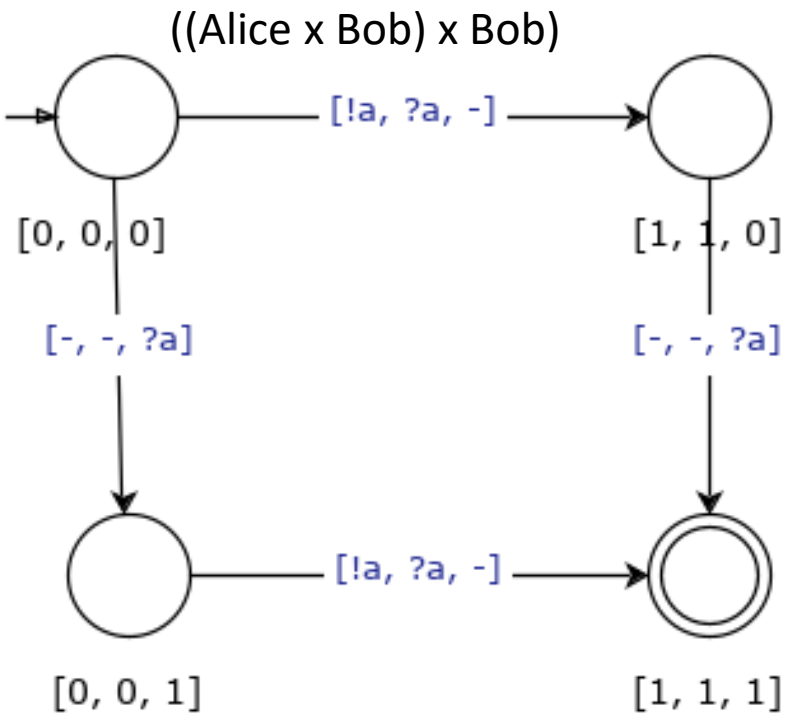


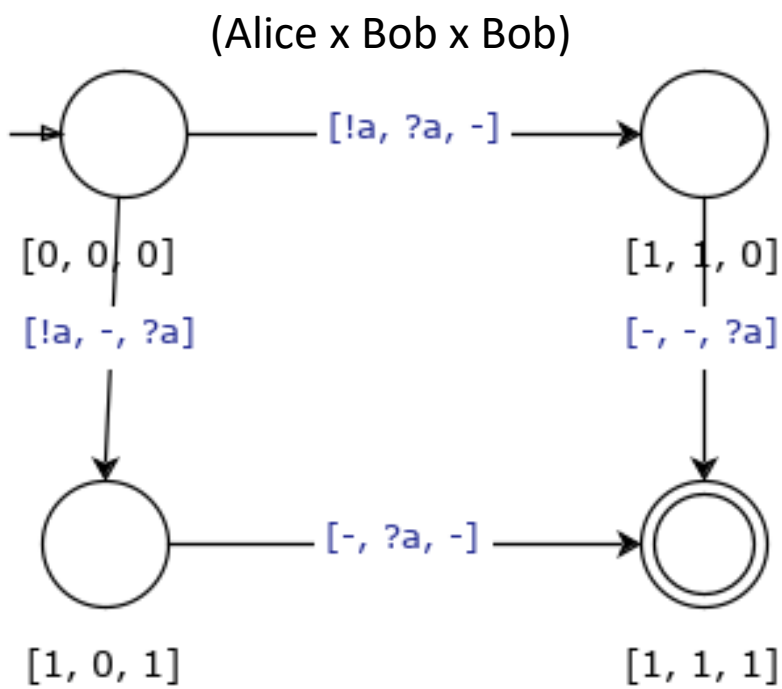
Compositionality, “Partiality”





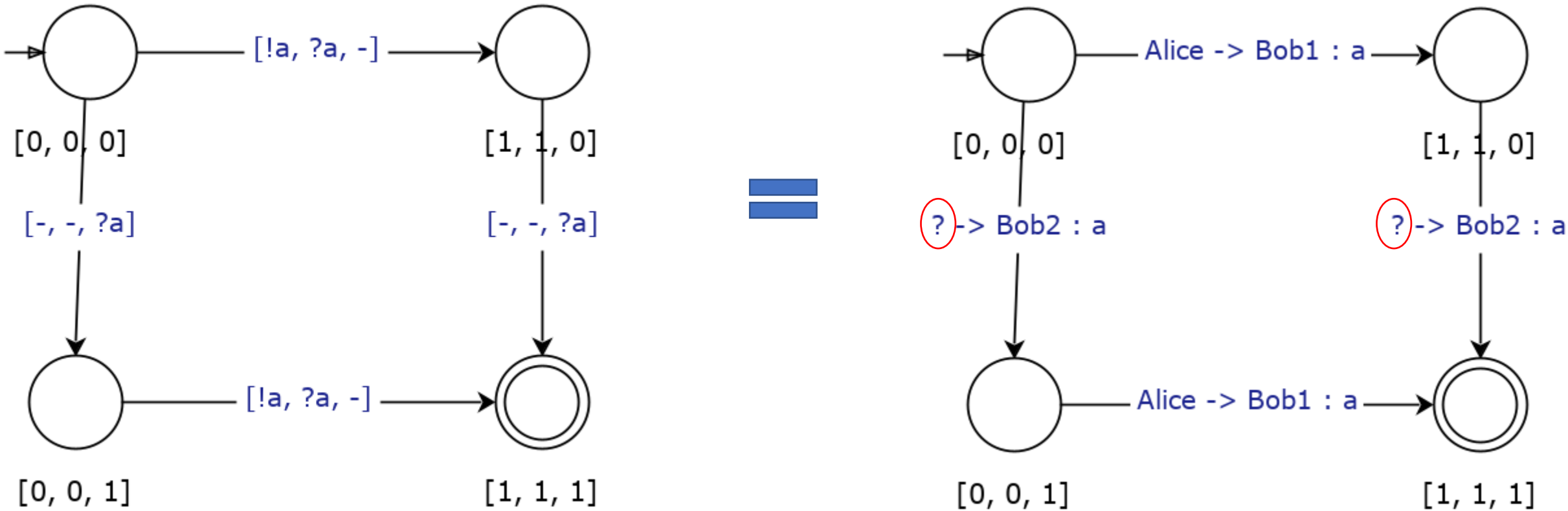






(Alice x Bob)

((Alice x Bob_1) x Bob_2)



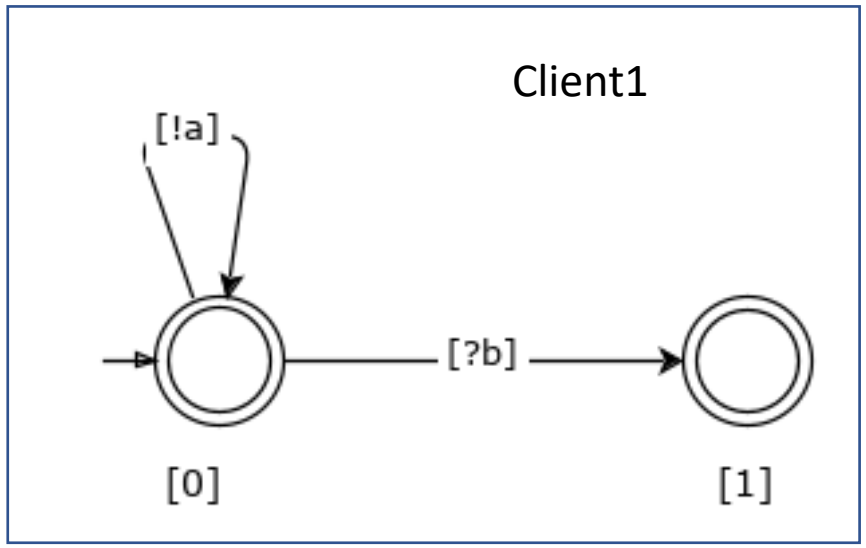
Orchestration Synthesis

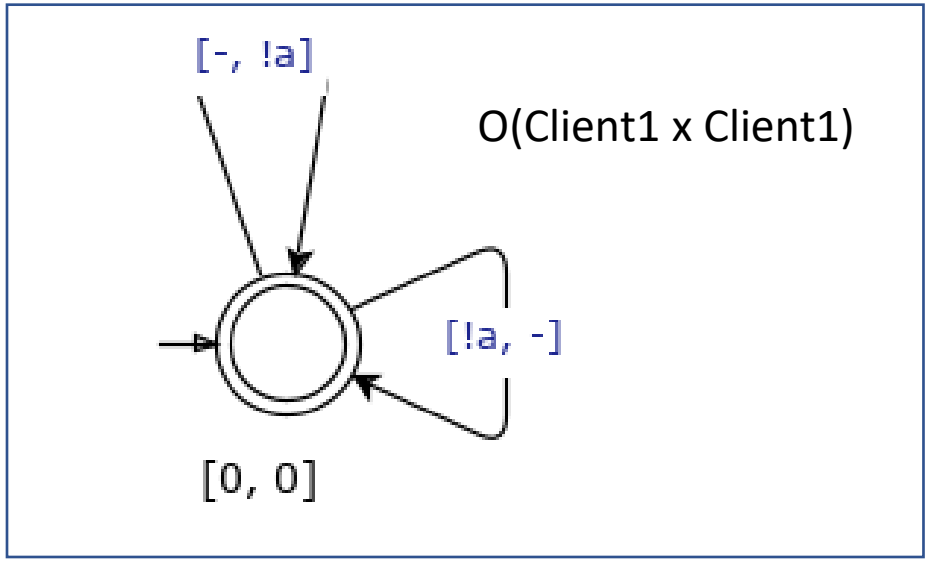
- The orchestration is computed using the synthesis of the most permissive controller (mpc) from Supervisory Control Theory (SCT) for discrete event systems
- The orchestrator adapts already existing services by blocking some of their actions to enforce *agreement* among the parties
  - (!a,?a,-)(!a,-,-) trace is in agreement (all requests are matched)
  - (!a,?a,-)**(-,?a,-)** trace <u>not</u> in agreement ((-,?a,-) is a request label)
- The orchestrator is *abstracted* away
- The synthesised orchestration assumes the presence of an orchestrator invoking the actions of the services

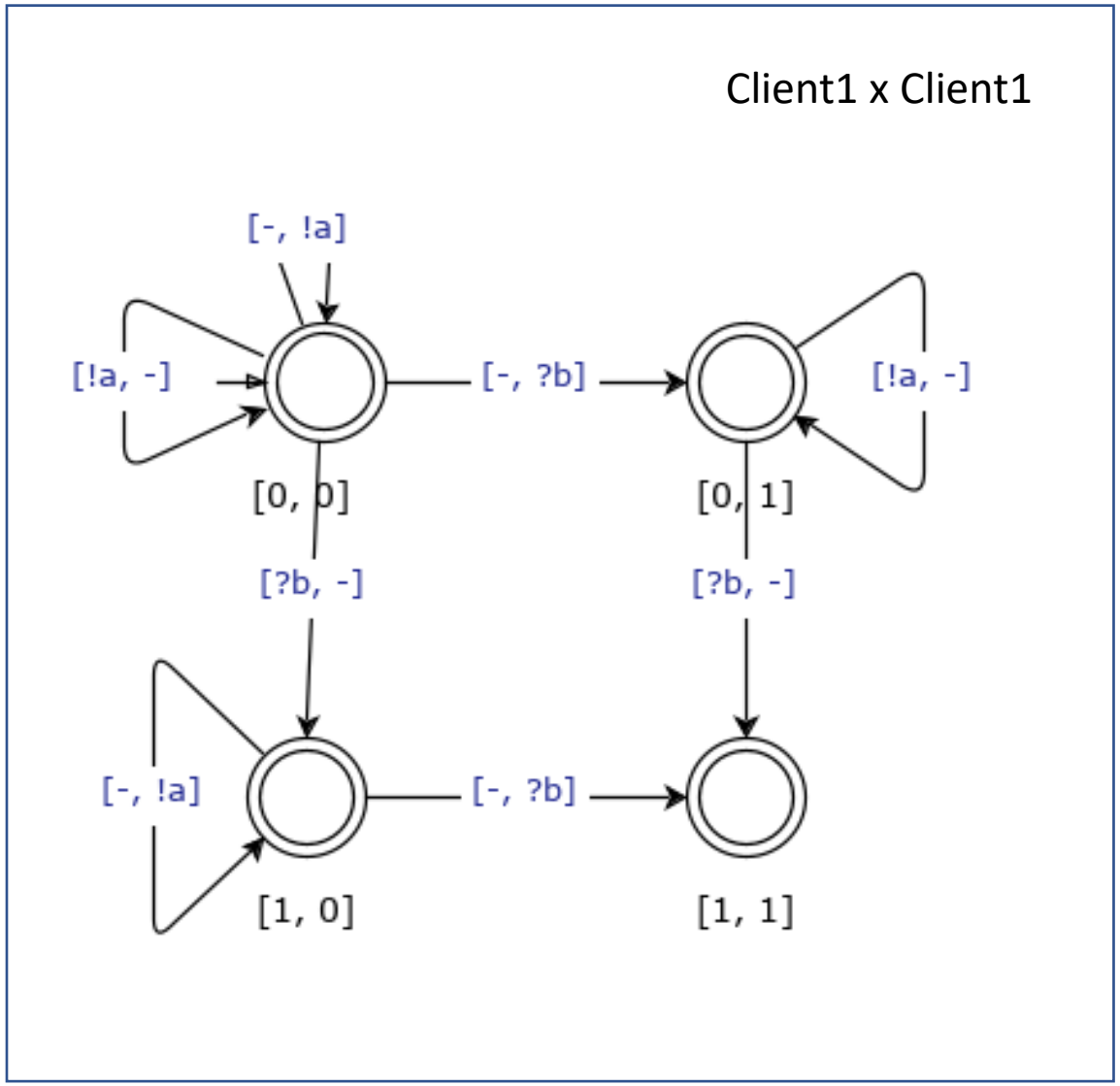
Properties of the orchestration

- The synthesis computes a refinement of the composition
  - Minimal fixed point algorithm
- SCT distinguishes between controllable and uncontrollable actions
- Only controllable actions can be removed (*controllability*)
- Final states (a.k.a. marked states) are always reachable (*non-blocking*)
- Forbidden states are never traversed (*safety*)
  - States with outgoing uncontrollable request transitions,
    - We are enforcing *agreement*,
  - Dangling states (i.e., unreachable)
- Minimal intervention of the orchestrator (*maximally permissive*)

Example 1 (requests are optional)

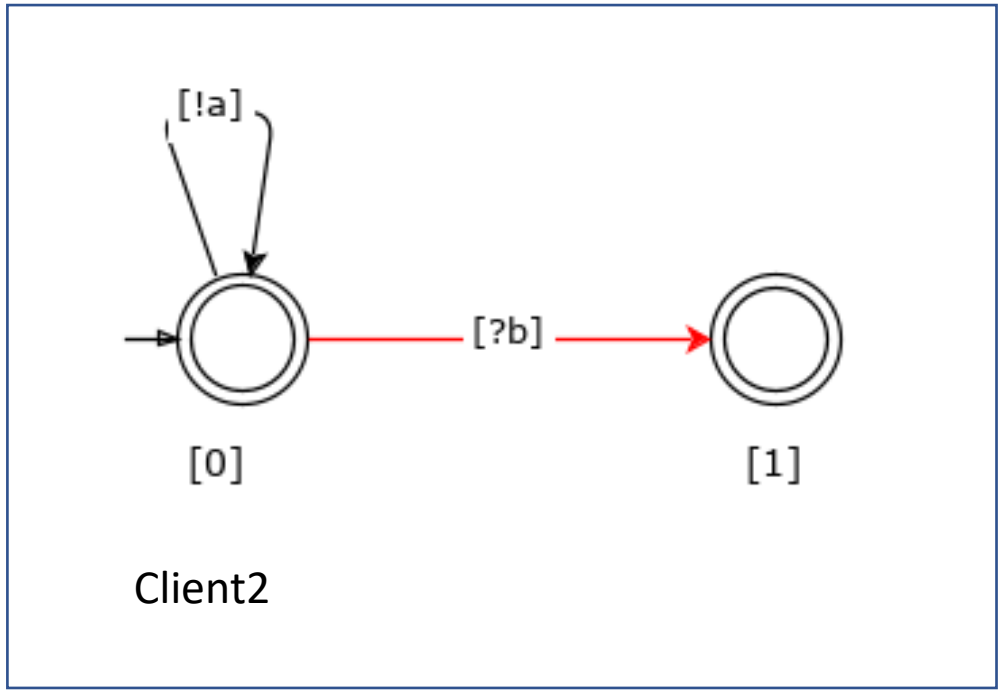


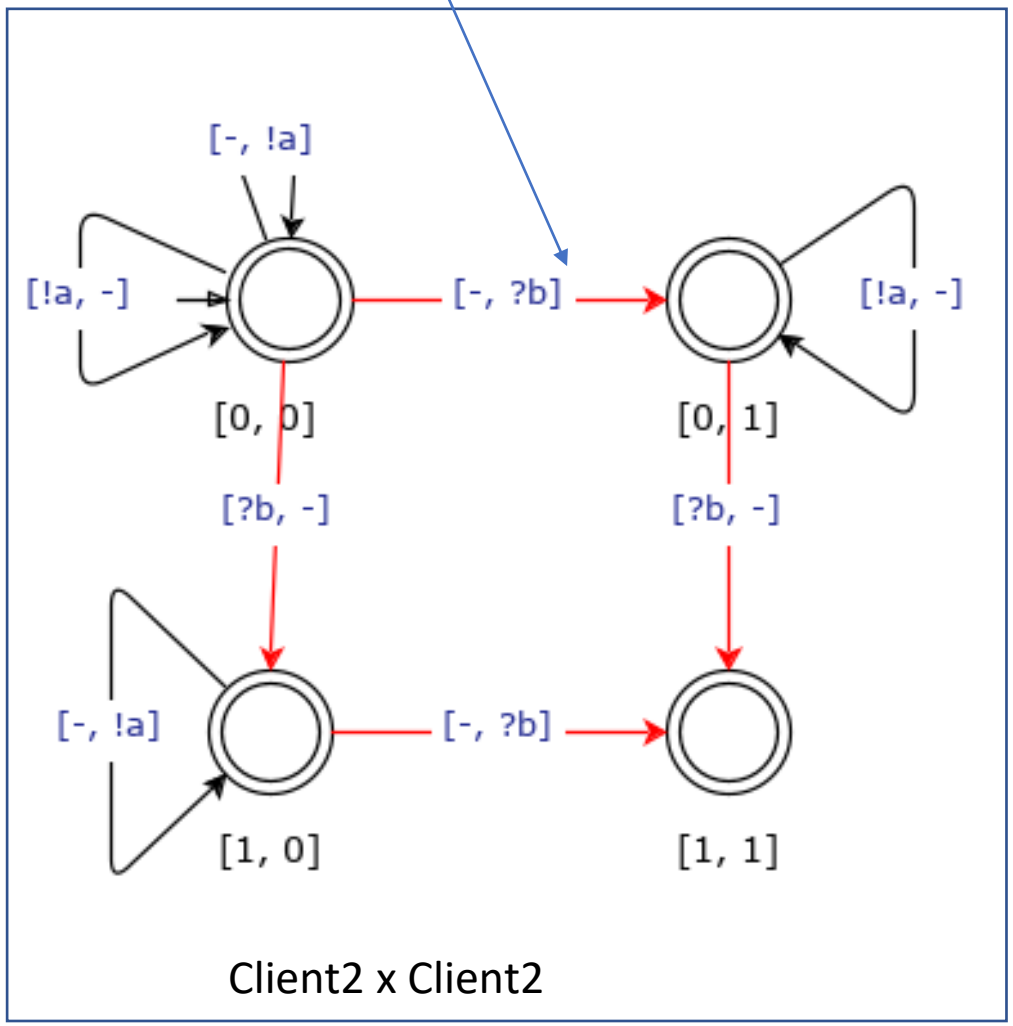




Example 2 (requests are necessary)

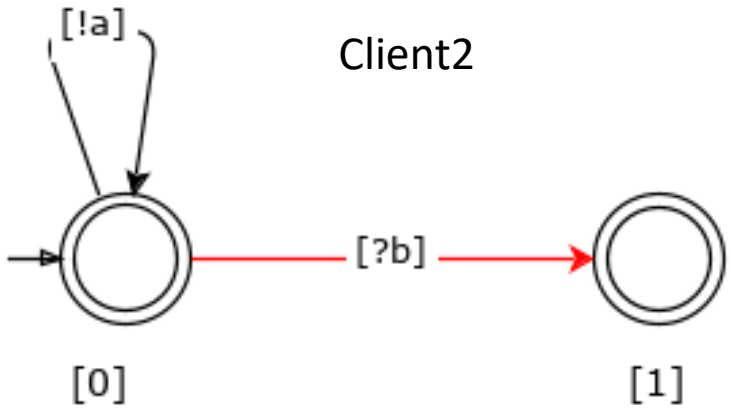
- The necessary request [?b] is *uncontrollable*
- O(Client2 x Client2) is empty, the initial state is *forbidden*!



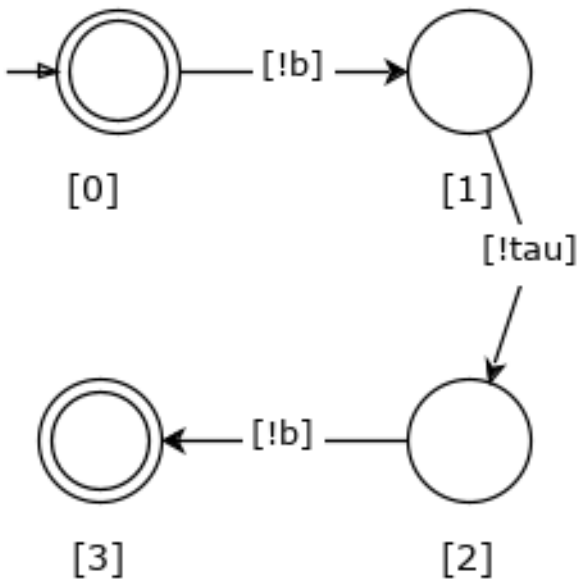


Example 3

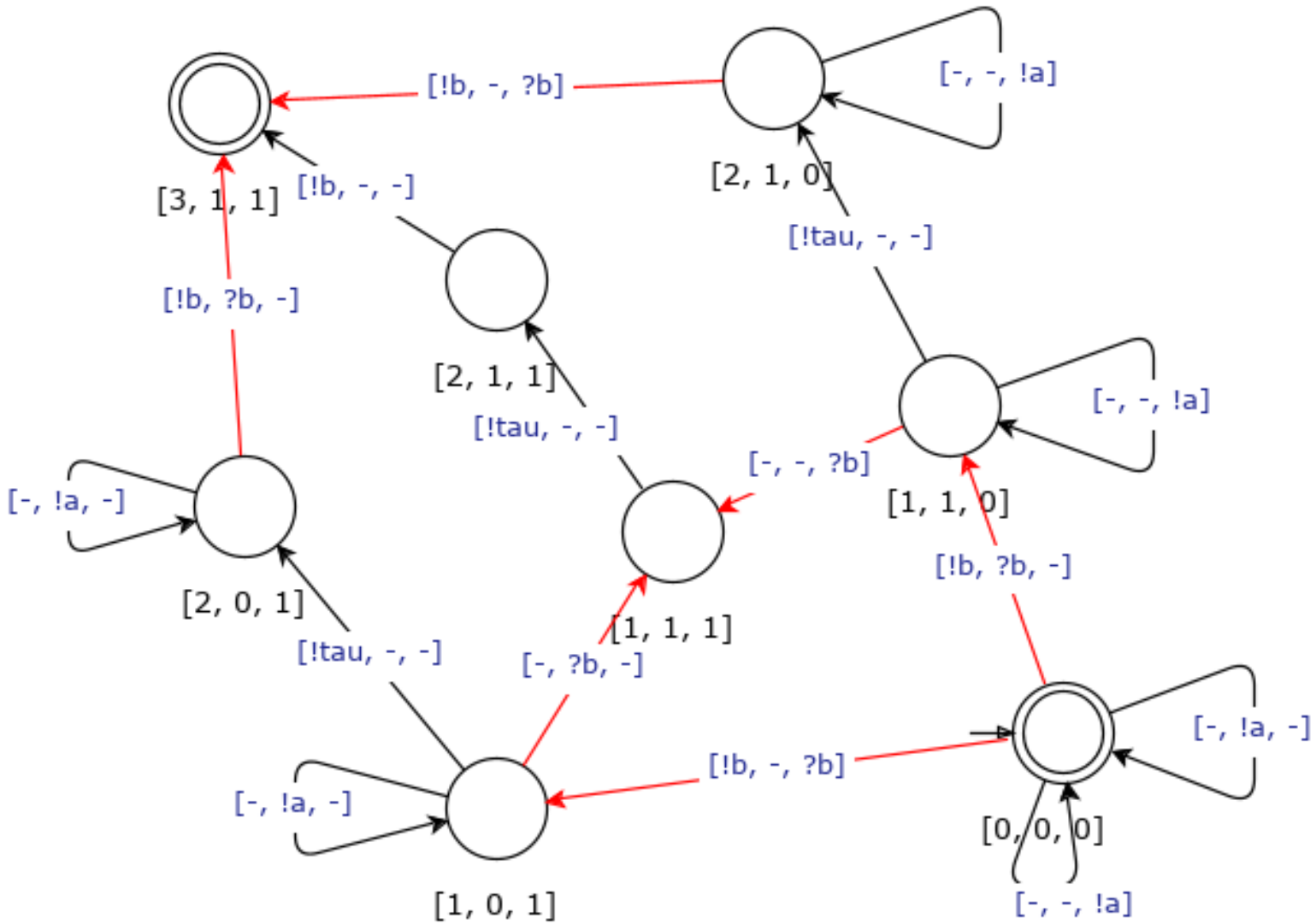
Client2



Server

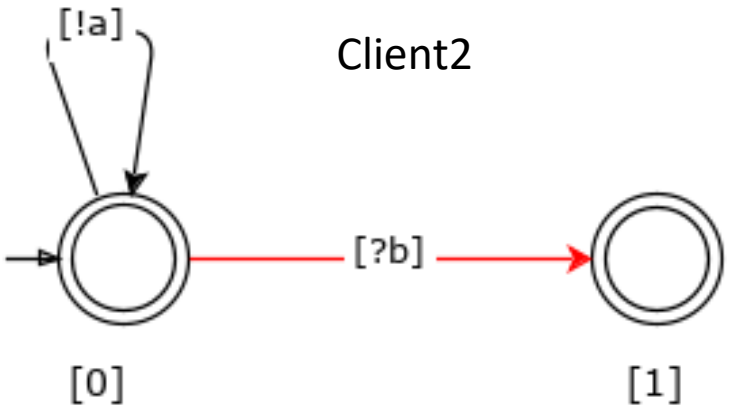


Server x Client2 x Client2

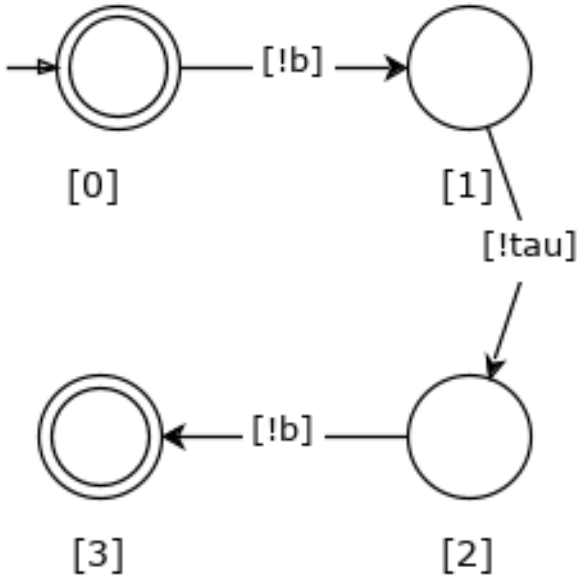


Example 3

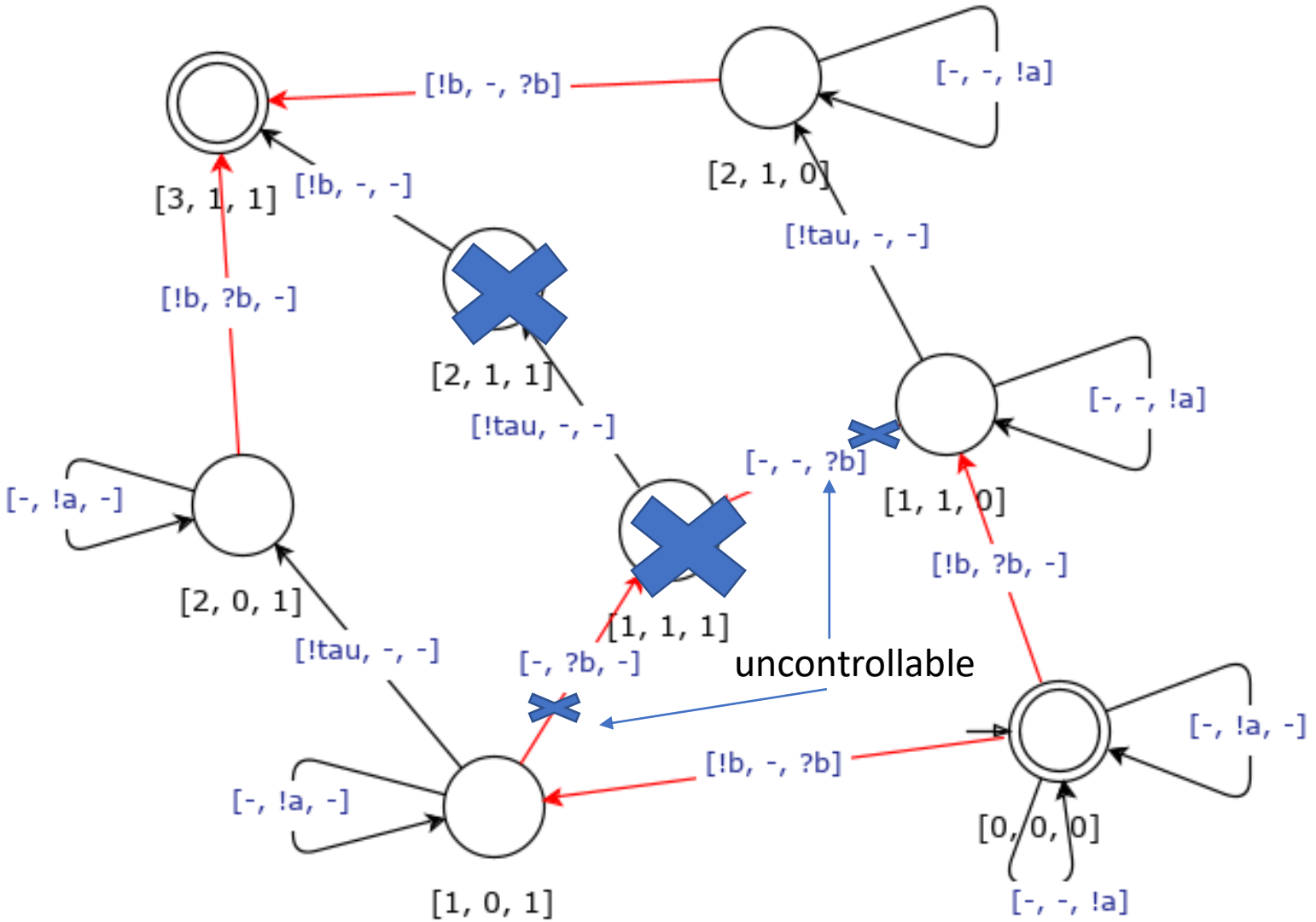
Client2



Server

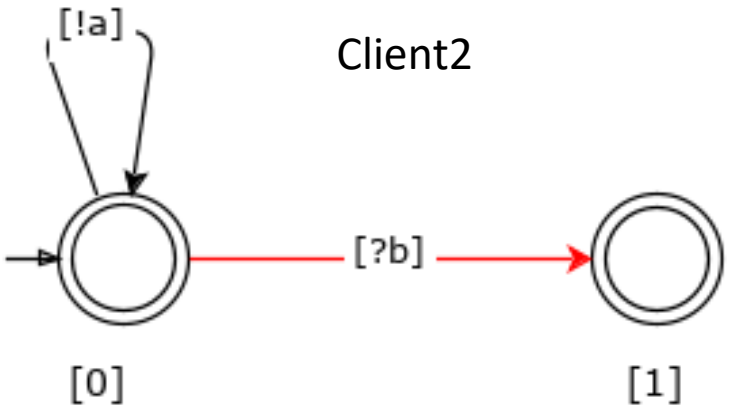


O(Server x Client2 x Client2) first iteration

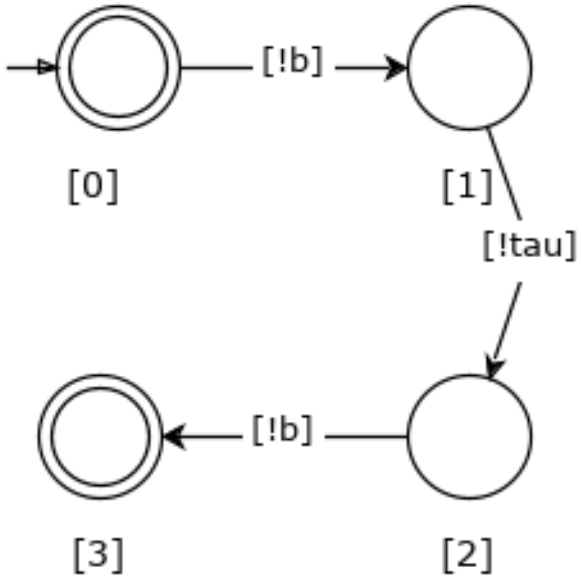


Example 3

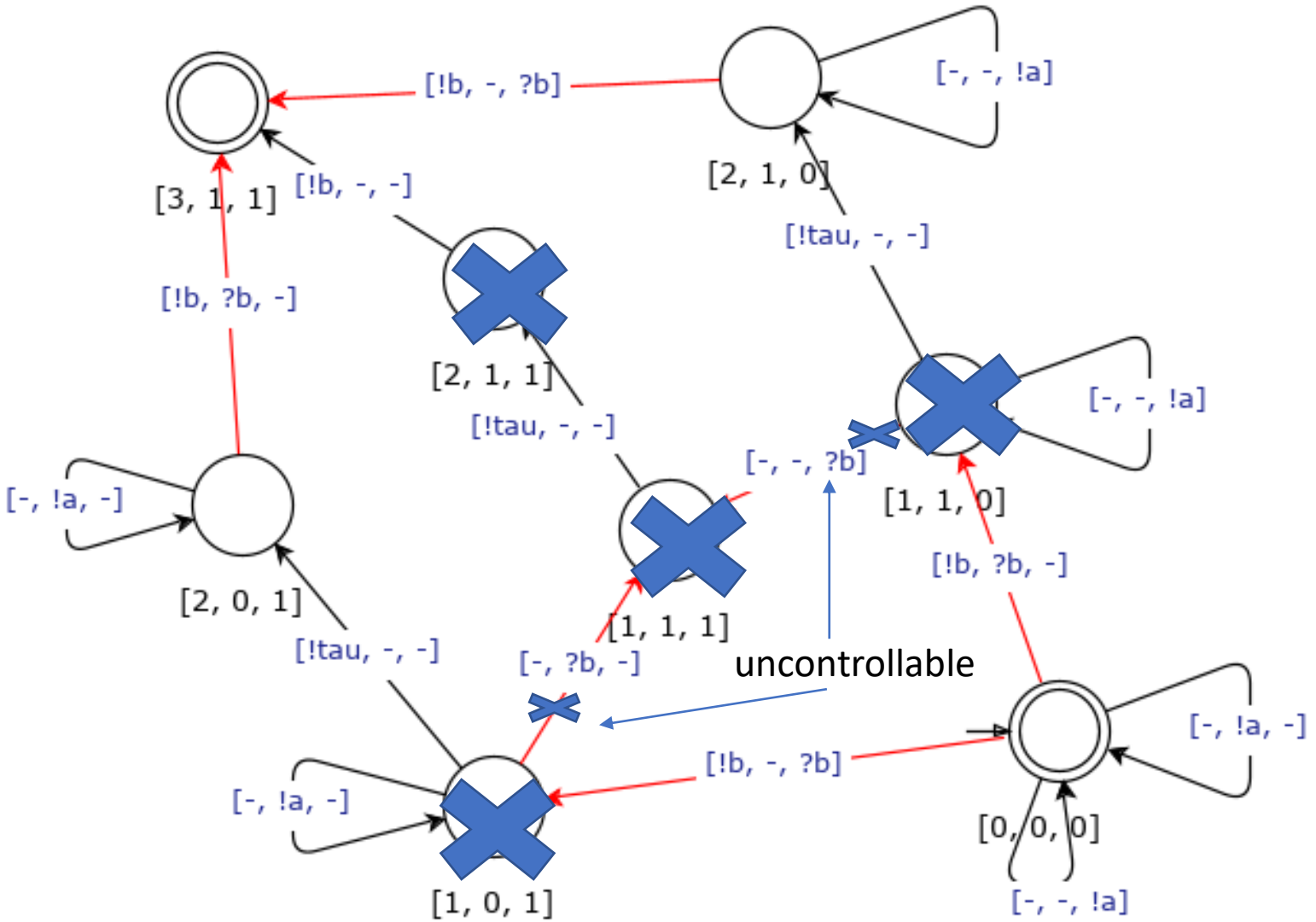
Client2



Server

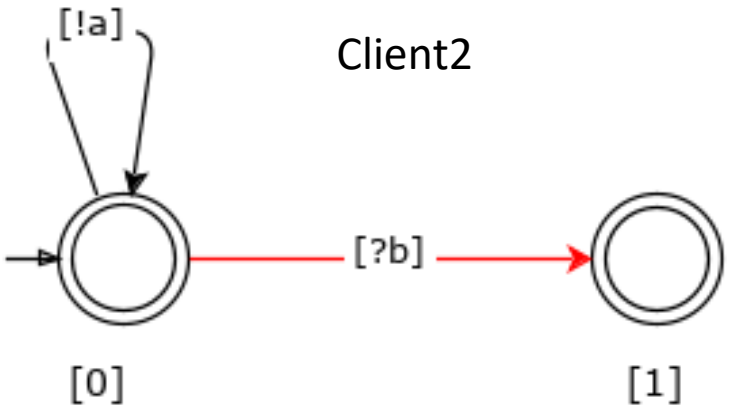


O(Server x Client2 x Client2) first iteration

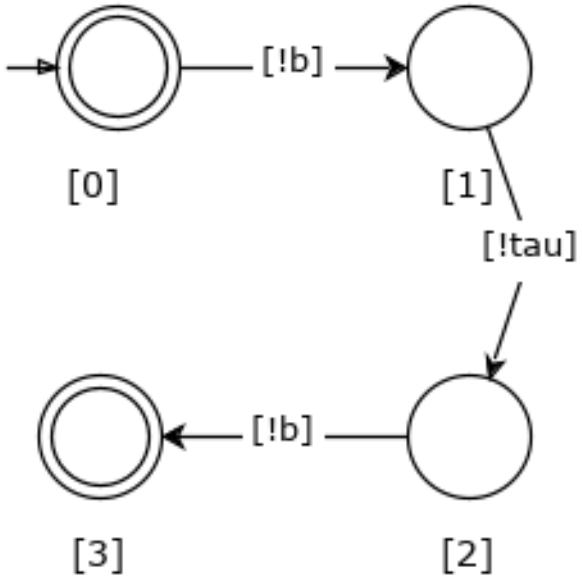


Example 3

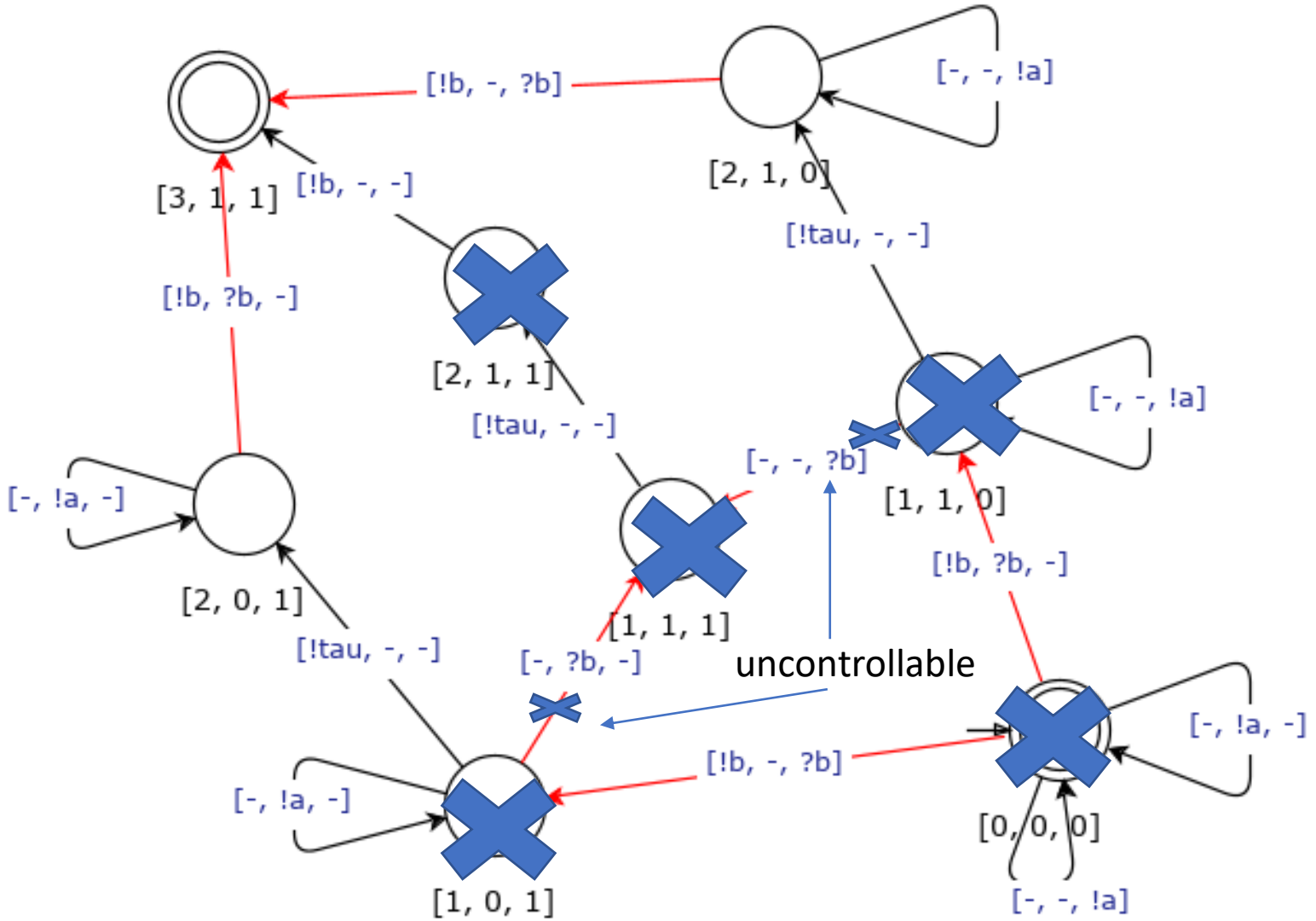
Client2



Server

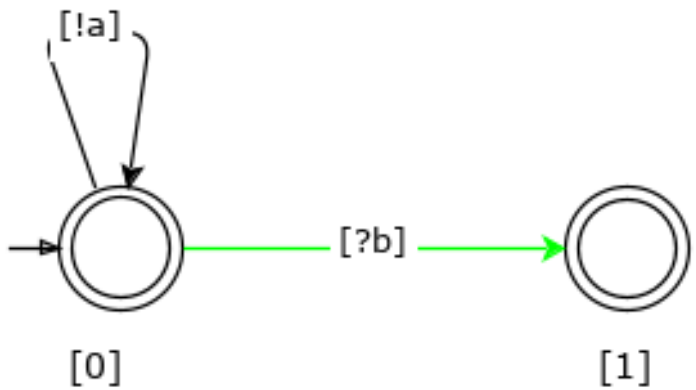


O(Server x Client2 x Client2) fixpoint

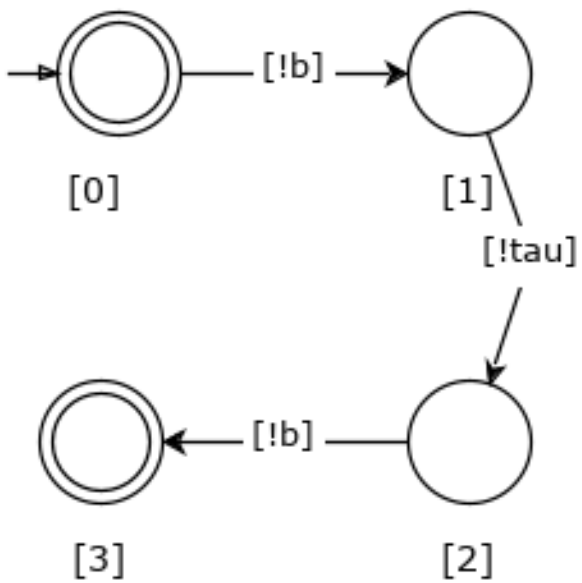


Example 4

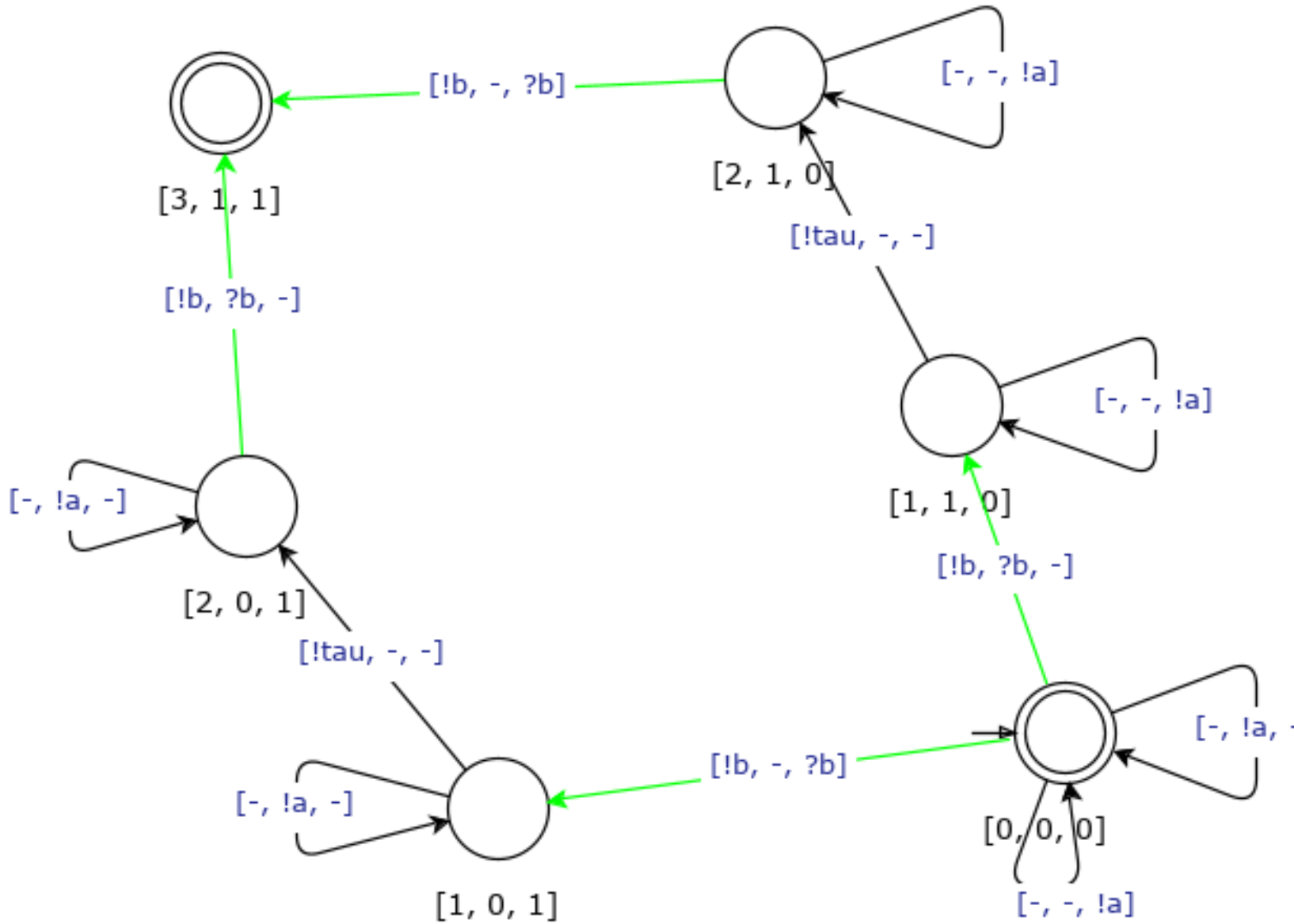
Client2



Server

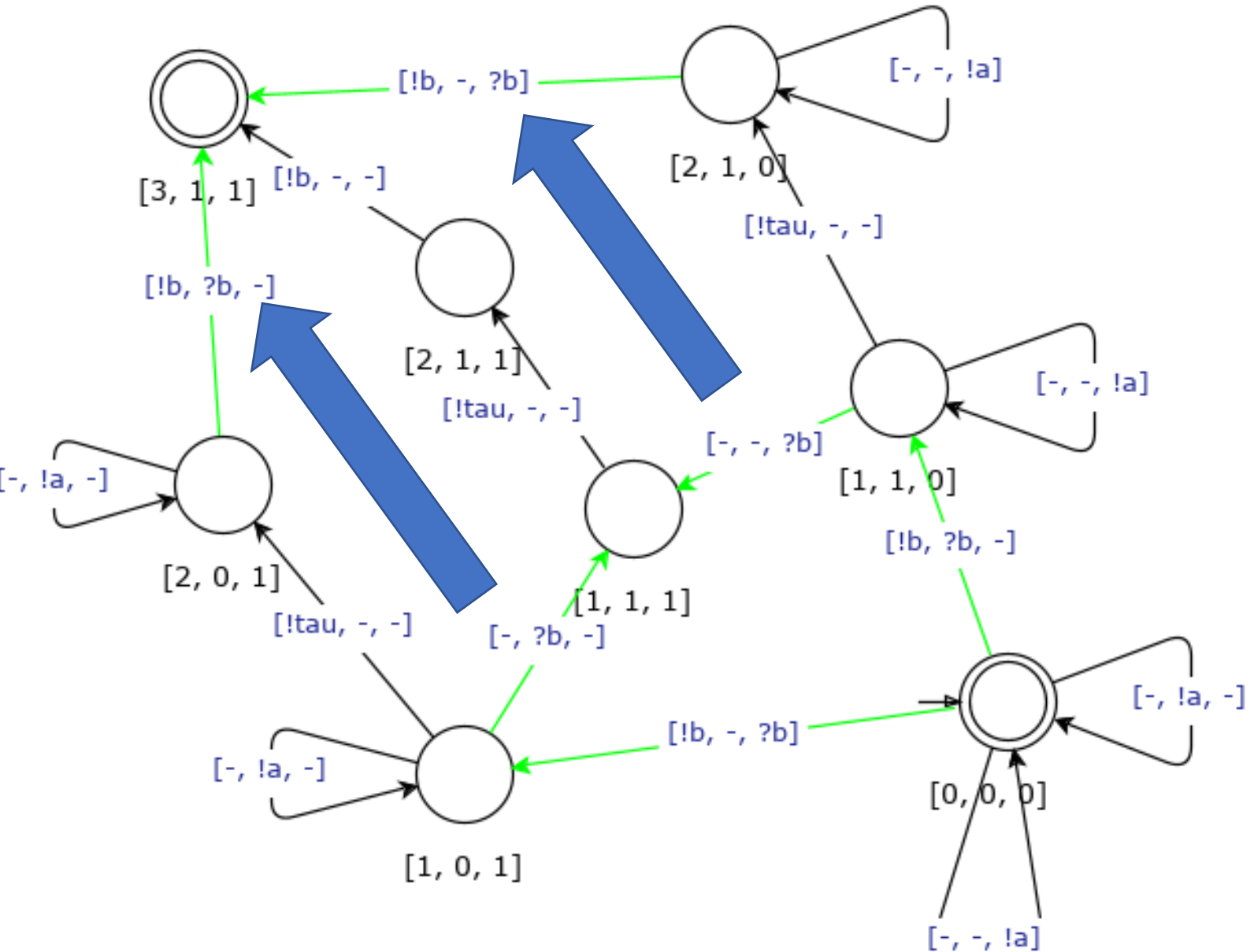


O(Server x Client2 x Client2)



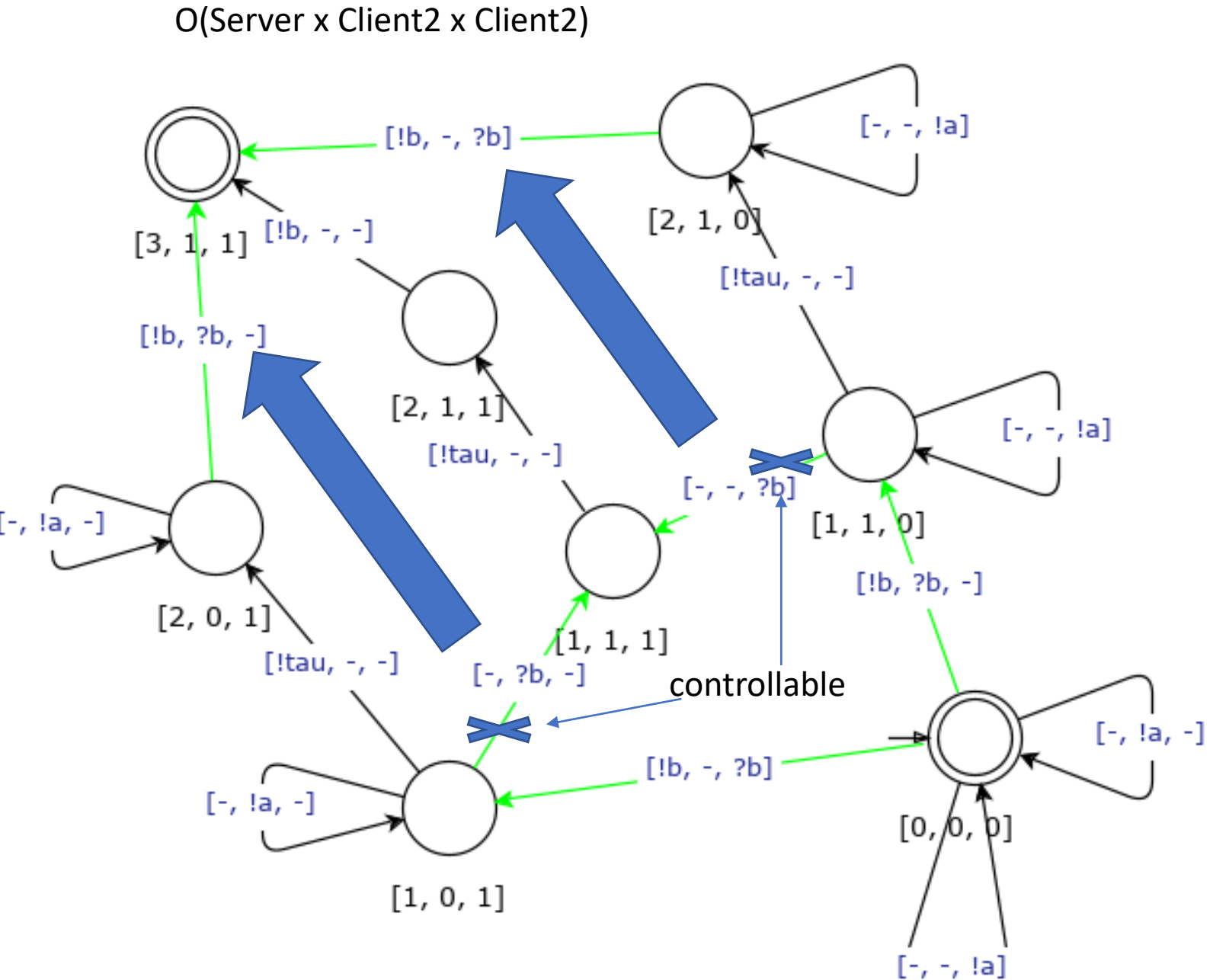
Semi-controllability

O(Server x Client2 x Client2)



Controllable **if** there *exists* another transition matching the same request of the same offerer performed from the same internal state, uncontrollable otherwise.

Semi-controllability



Controllable **if** there *exists* another transition matching the same request of the same offerer performed from the same internal state, uncontrollable otherwise.

Contract Automata Runtime Environment

- CARE is a middleware recently introduced for realizing applications specified via contract automata
- It provides an implementation of an *orchestrator*
- Two aspects to concretise in CARE are
  - Implementation of actions
  - Implementation of choices
    - Abstracted in contract automata

- Basile, D. and ter Beek, M.H., 2023, March. A runtime environment for contract automata. In *FM2023*
- https://github.com/contractautomataproject (tools: CATLib, CARE, CATApp)

Example 5

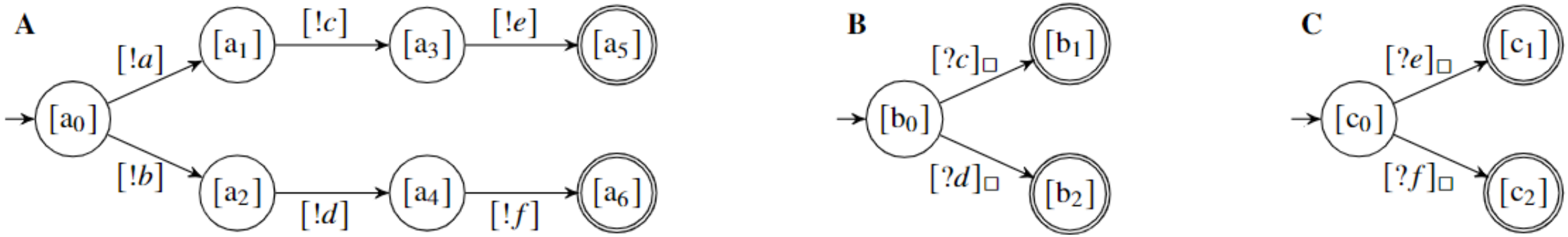


Figure 4: Contracts of **A**lice, **B**ob and **C**arl

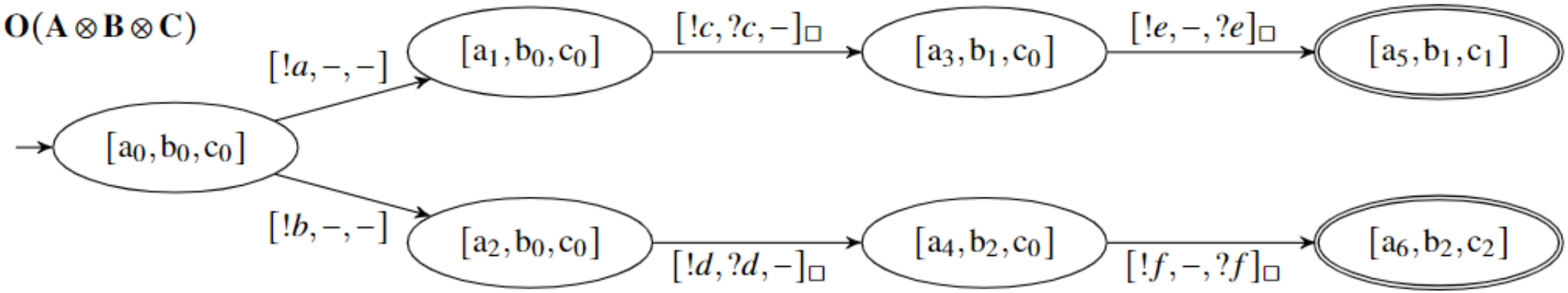
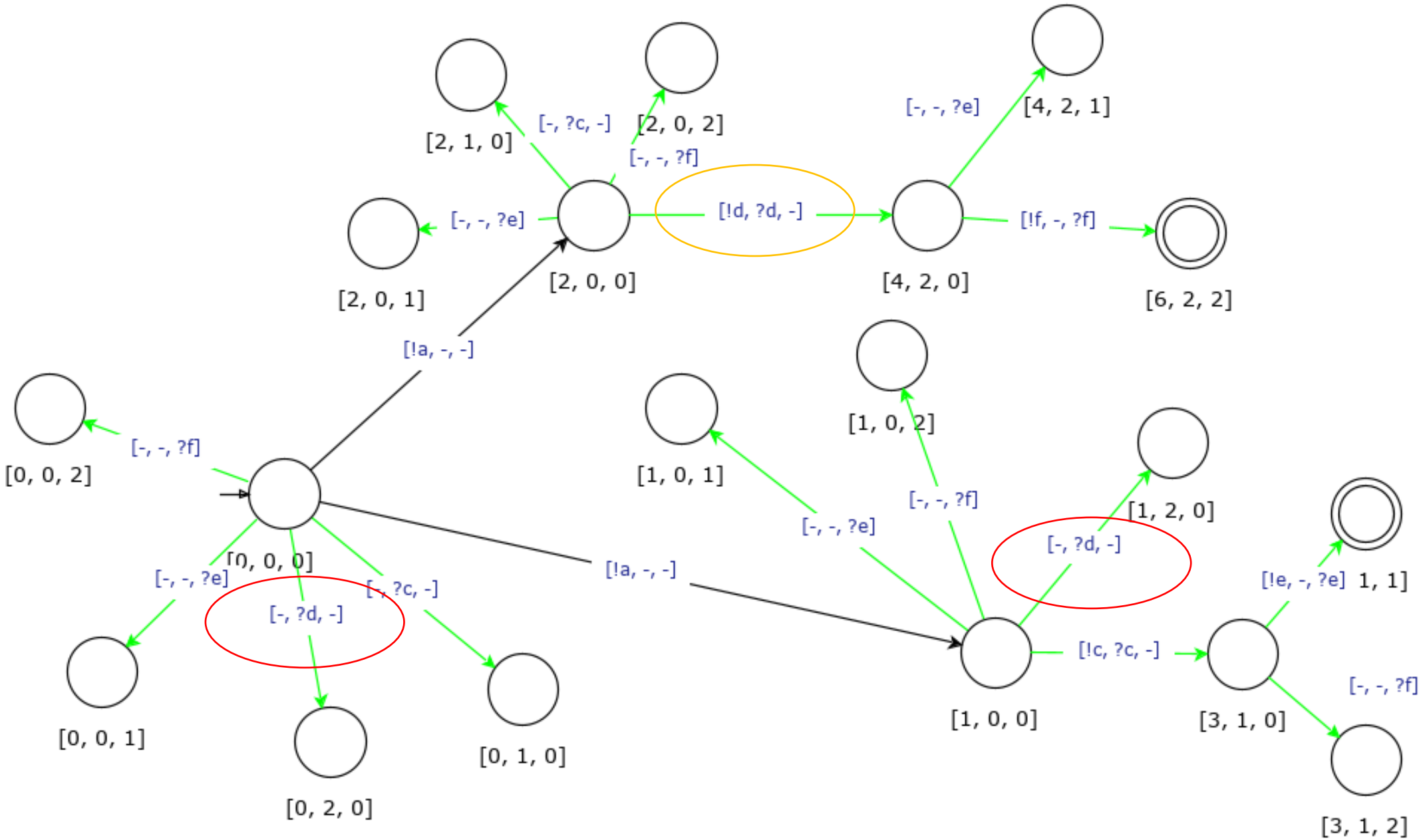


Figure 5: Orchestration $\mathbf{O}(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})$ of $\mathbf{A}\text{lice} \otimes \mathbf{B}\text{ob} \otimes \mathbf{C}\text{arl}$

Refined Semi-controllability also require that [2, 0 ,0] is reachable from [1, 0, 0]



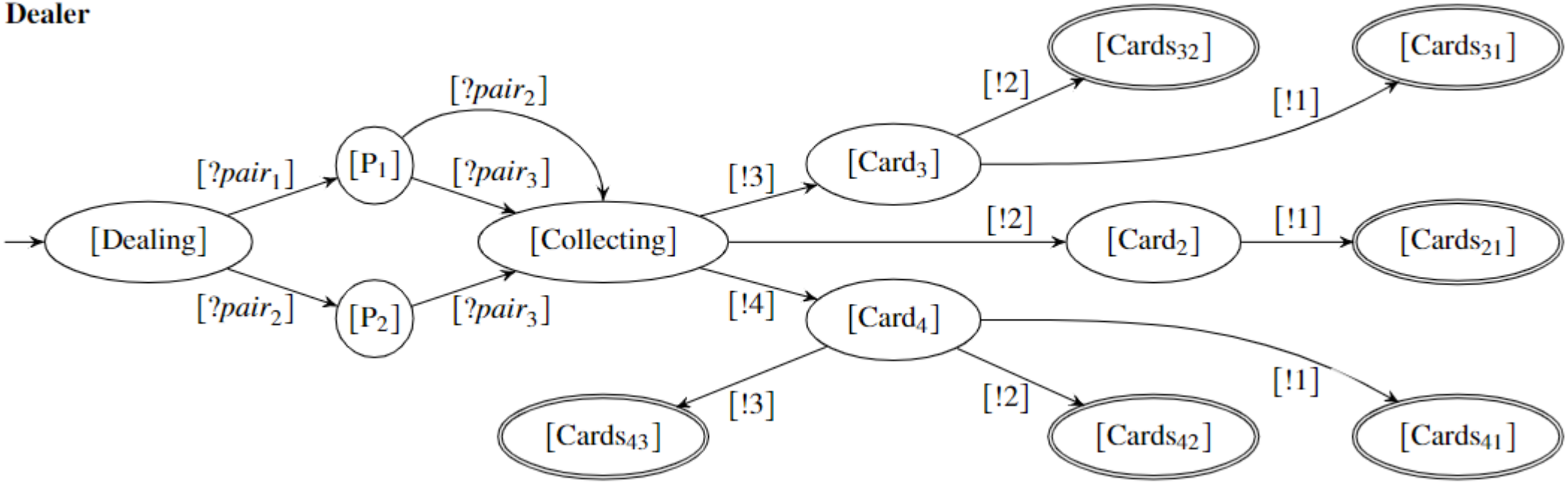
Intuition of necessary service requests

- A service *internally* decides whether to perform a necessary request
- The orchestrator *controls* the scheduling of the requests

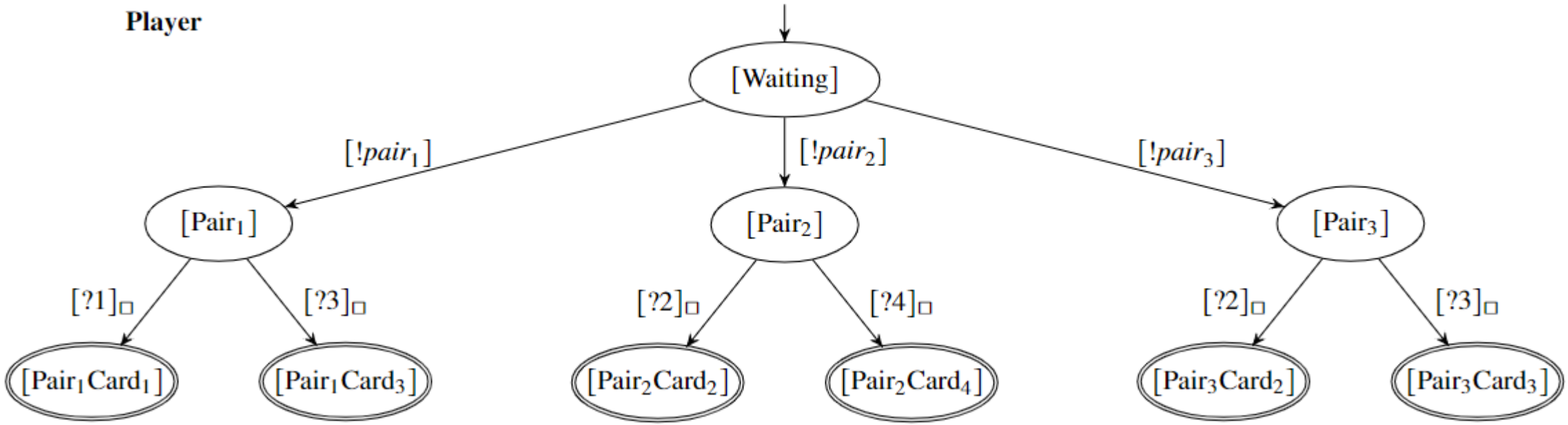
Card dealer example

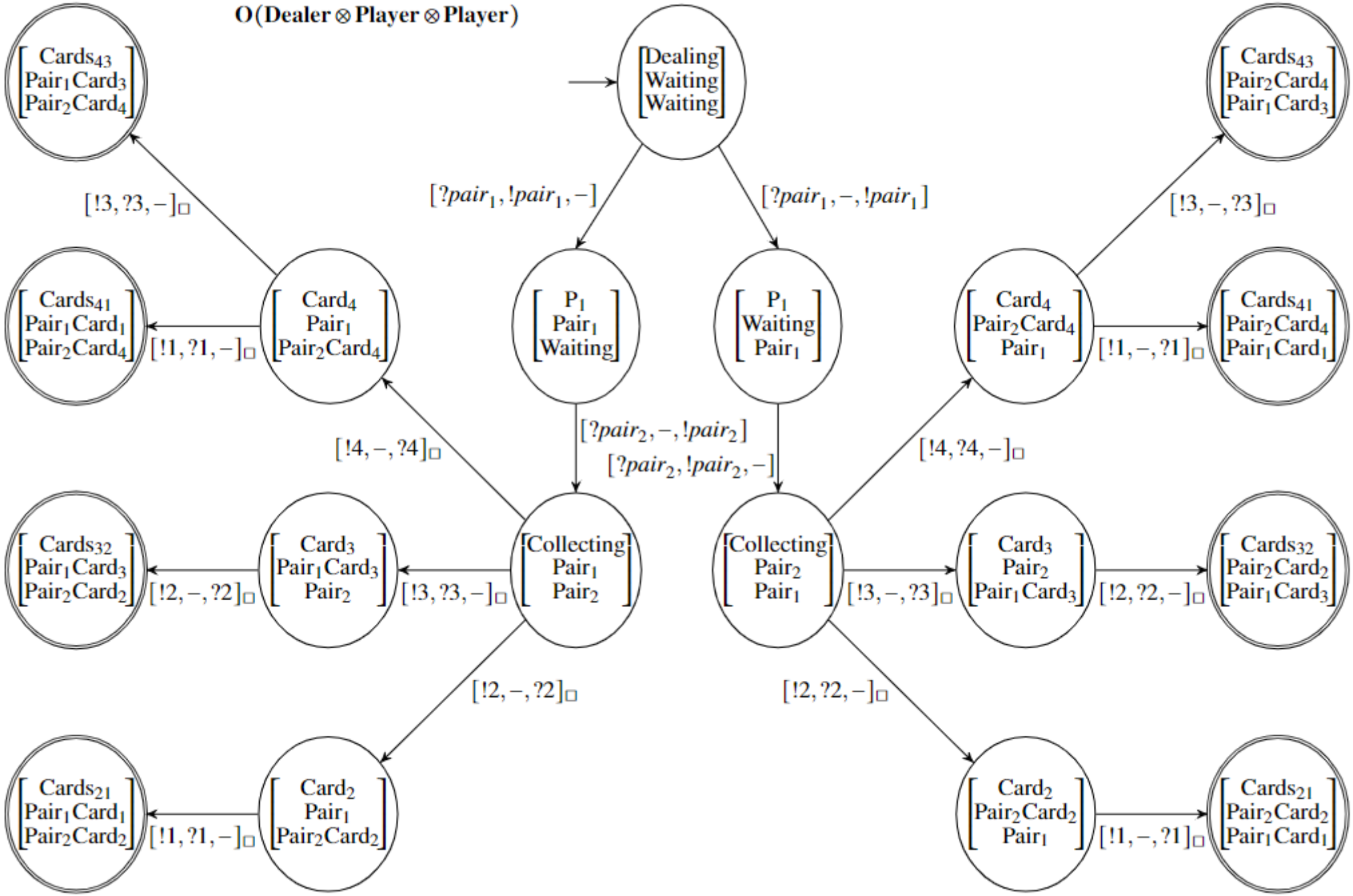
- Each player receives a pair of cards selected by the dealer from
  - (1,3), (2,4), (2,3)
- Each player picks one of the two cards
- The dealer shall collect the selected cards in descending order
- Strategy of the dealer:
  - Deal in no particular order the pairs (1,3) and (2,4)

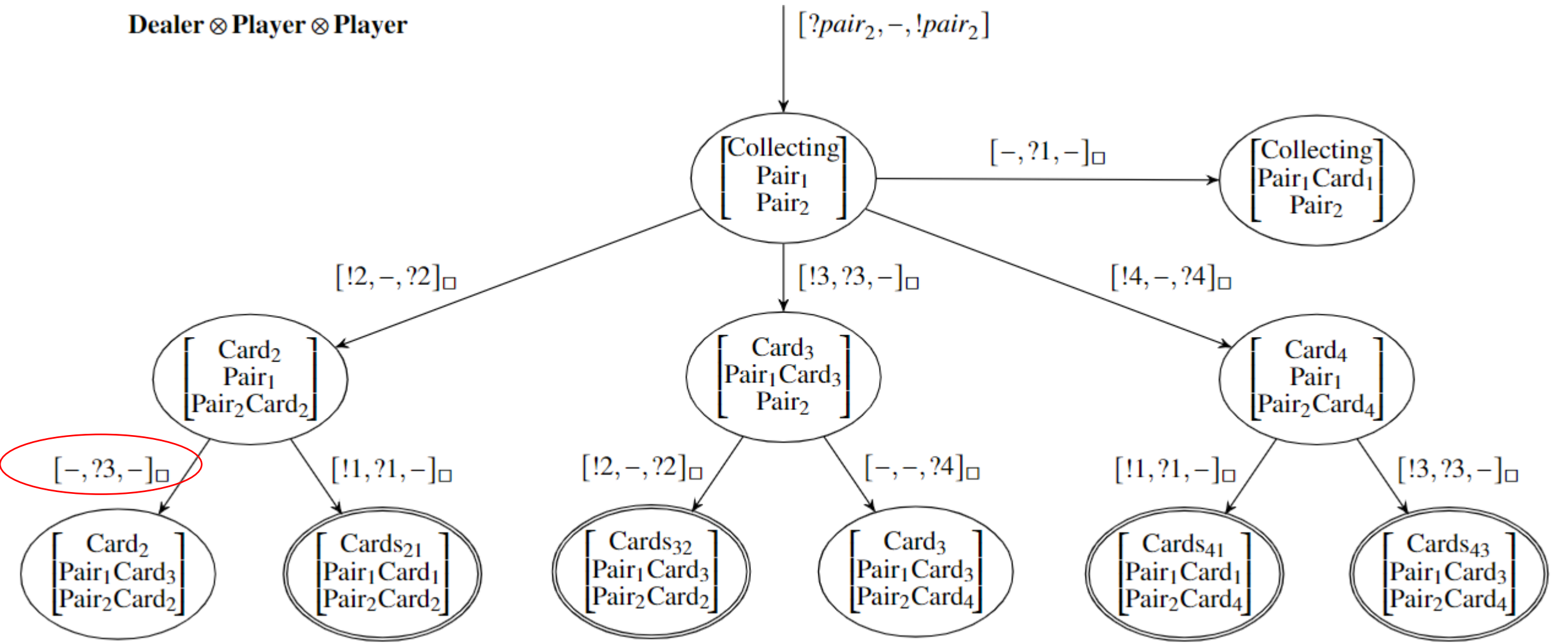
Dealer

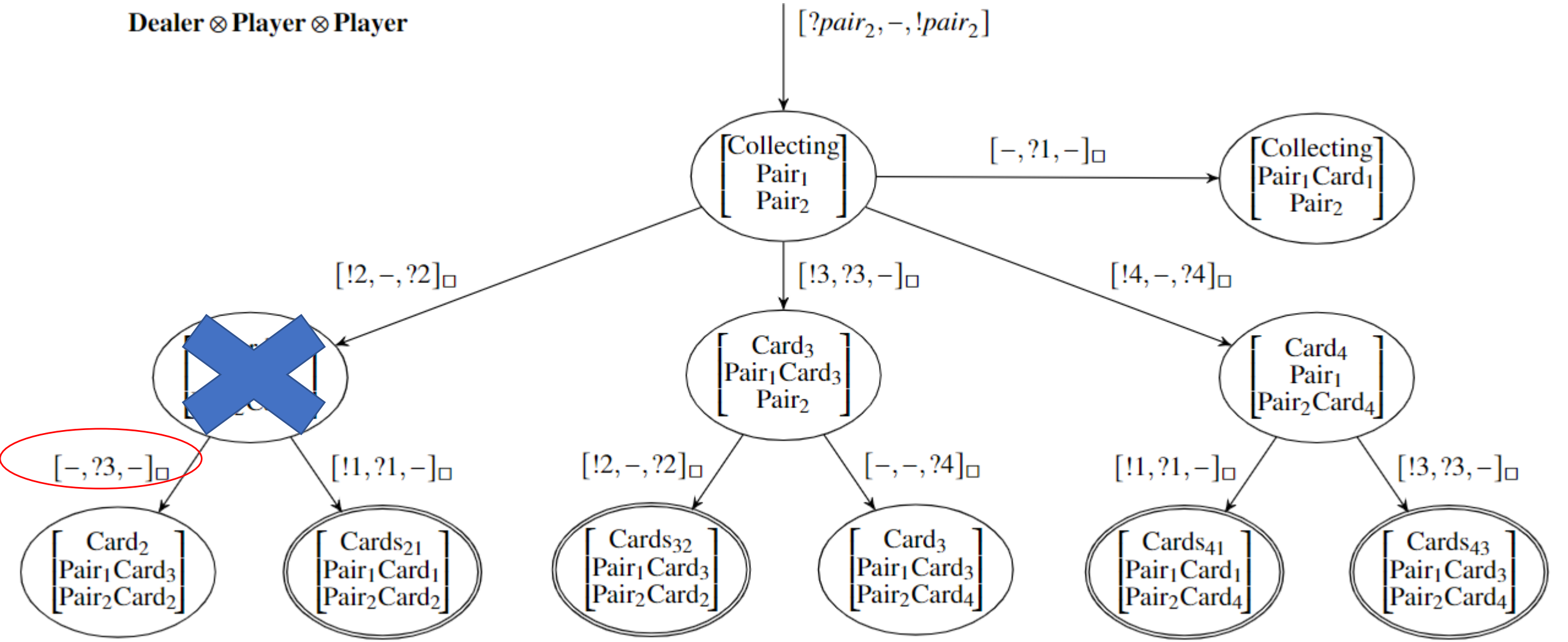


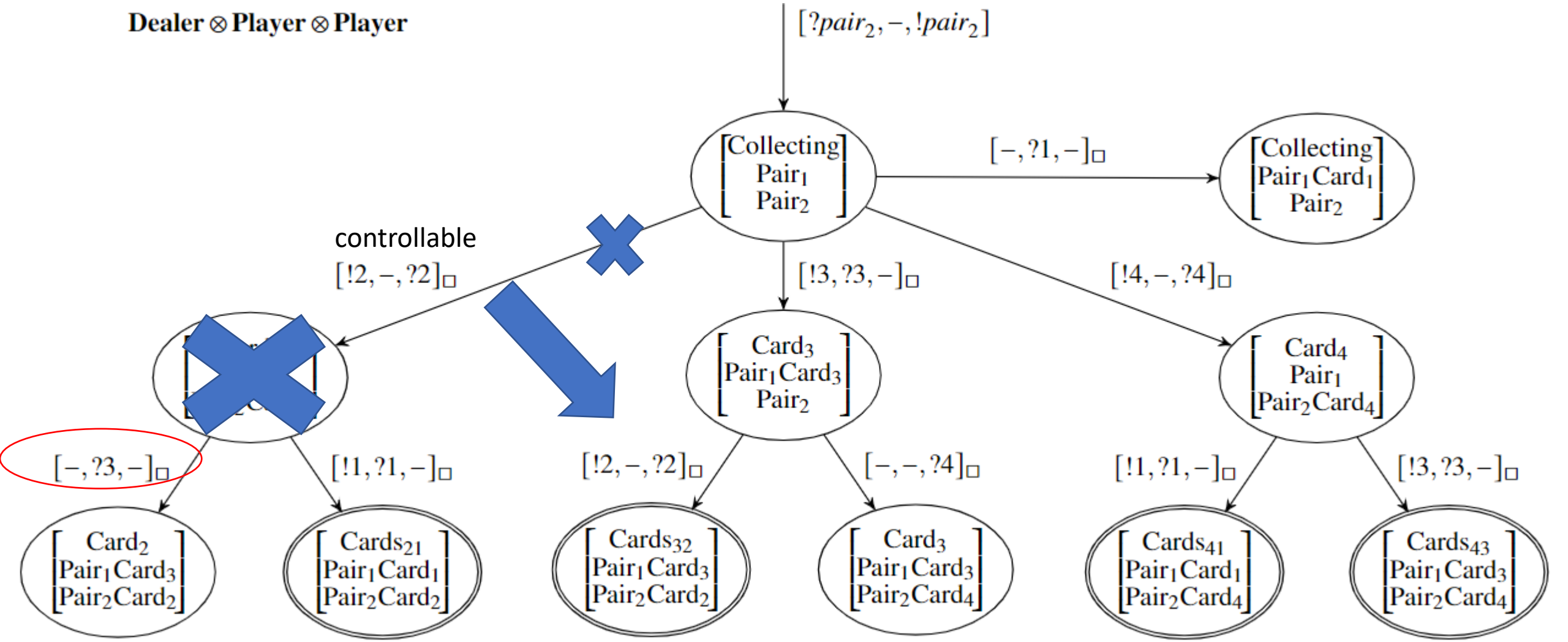
Player

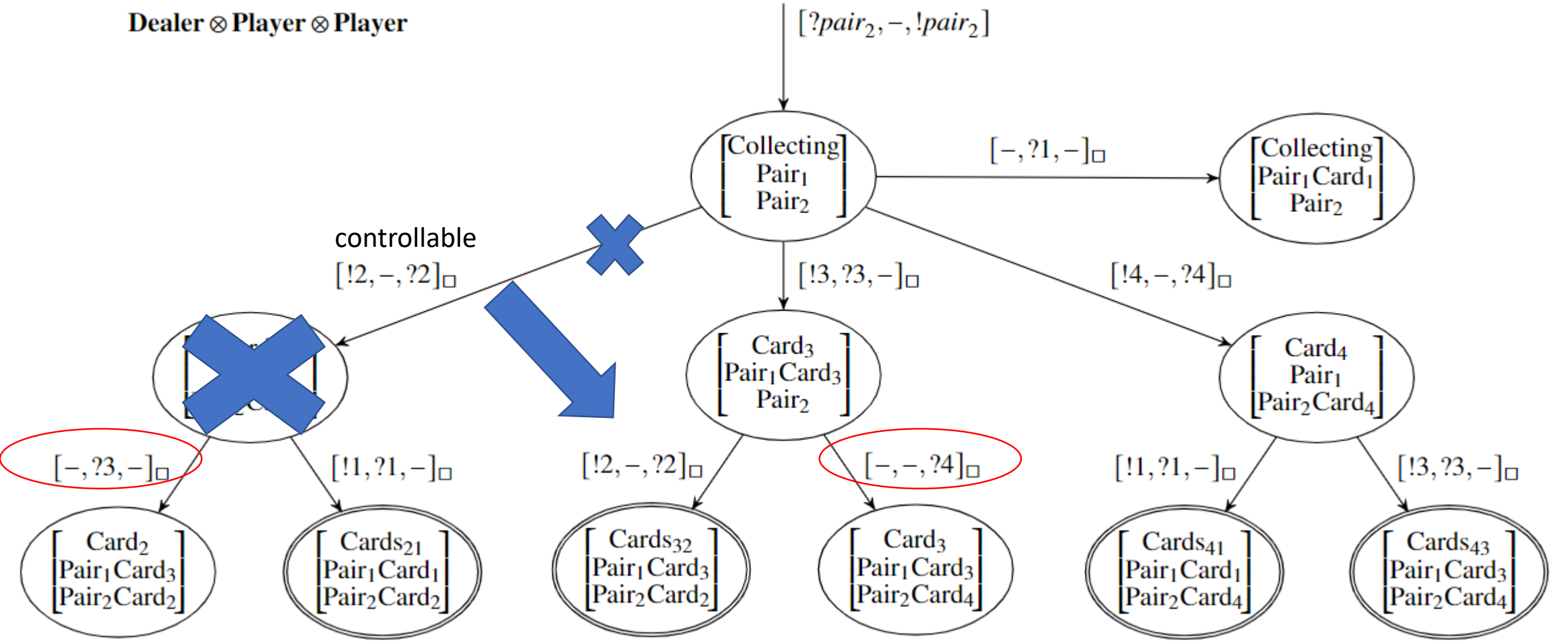


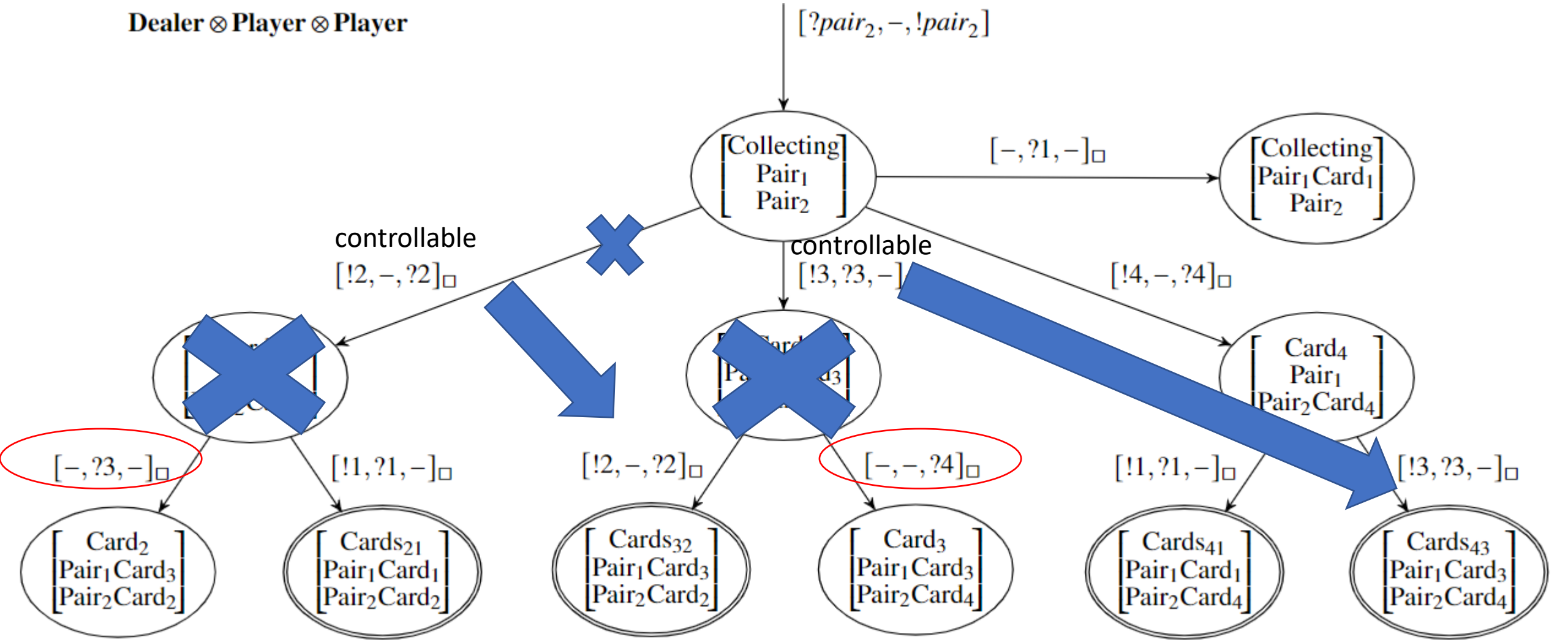

O(Dealer ⊗ Player ⊗ Player)
Dealing Waiting Waiting
[?pair1, !pair1, −]
[?pair1, −, !pair1]
P1 Pair1 Waiting
P1 Waiting Pair1
Card4 Pair1 Pair2Card4
[!3, ?3, −]□
Cards43 Pair1Card3 Pair2Card4
Cards41 Pair1Card1 Pair2Card4
[!1, ?1, −]□
[!3, −, ?3]□
Cards43 Pair2Card4 Pair1Card3
Card4 Pair2Card4 Pair1
[!1, −, ?1]□
Cards41 Pair2Card4 Pair1Card1
[?pair2, −, !pair2]
[?pair2, !pair2, −]
[!4, −, ?4]□
[!4, ?4, −]□
Collecting Pair1 Pair2
Collecting Pair2 Pair1
Cards32 Pair1Card3 Pair2Card2
Card3 Pair1Card3 Pair2
[!2, −, ?2]□
[!3, ?3, −]□
[!3, −, ?3]□
Card3 Pair2 Pair1Card3
[!2, ?2, −]□
Cards32 Pair2Card2 Pair1Card3
[!2, −, ?2]□
[!2, ?2, −]□
Cards21 Pair1Card1 Pair2Card2
Card2 Pair1 Pair2Card2
[!1, ?1, −]□
Card2 Pair2Card2 Pair1
[!1, −, ?1]□
Cards21 Pair2Card2 Pair1Card1

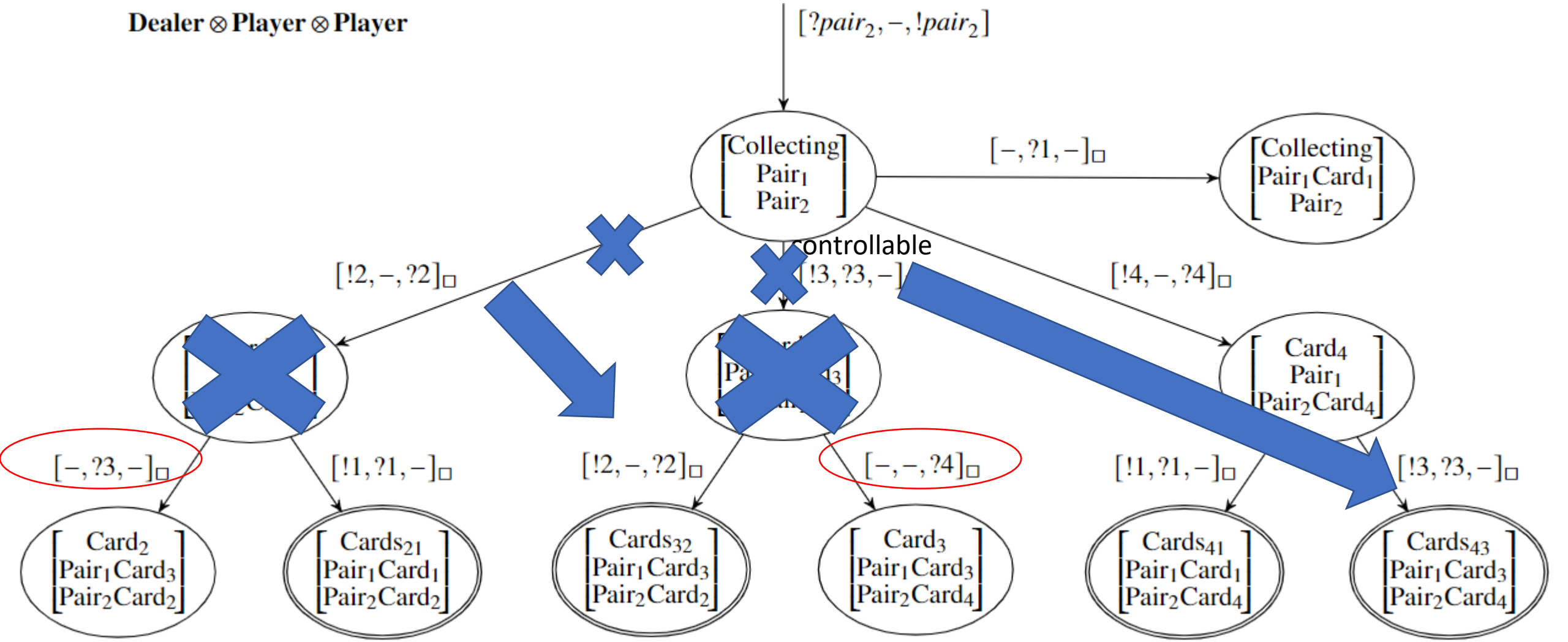

Dealer ⊗ Player ⊗ Player
[?pair2, −, !pair2]
[Collecting, Pair1, Pair2]
[−, ?1, −]□
[Collecting, Pair1 Card1, Pair2]
[!2, −, ?2]□
[!3, ?3, −]□
[!4, −, ?4]□
[Card2, Pair1, Pair2 Card2]
[Card3, Pair1 Card3, Pair2]
[Card4, Pair1, Pair2 Card4]
[−, ?3, −]□
[!1, ?1, −]□
[!2, −, ?2]□
[−, −, ?4]□
[!1, ?1, −]□
[!3, ?3, −]□
[Card2, Pair1 Card3, Pair2 Card2]
[Cards21, Pair1 Card1, Pair2 Card2]
[Cards32, Pair1 Card3, Pair2 Card2]
[Card3, Pair1 Card3, Pair2 Card4]
[Cards41, Pair1 Card1, Pair2 Card4]
[Cards43, Pair1 Card3, Pair2 Card4]

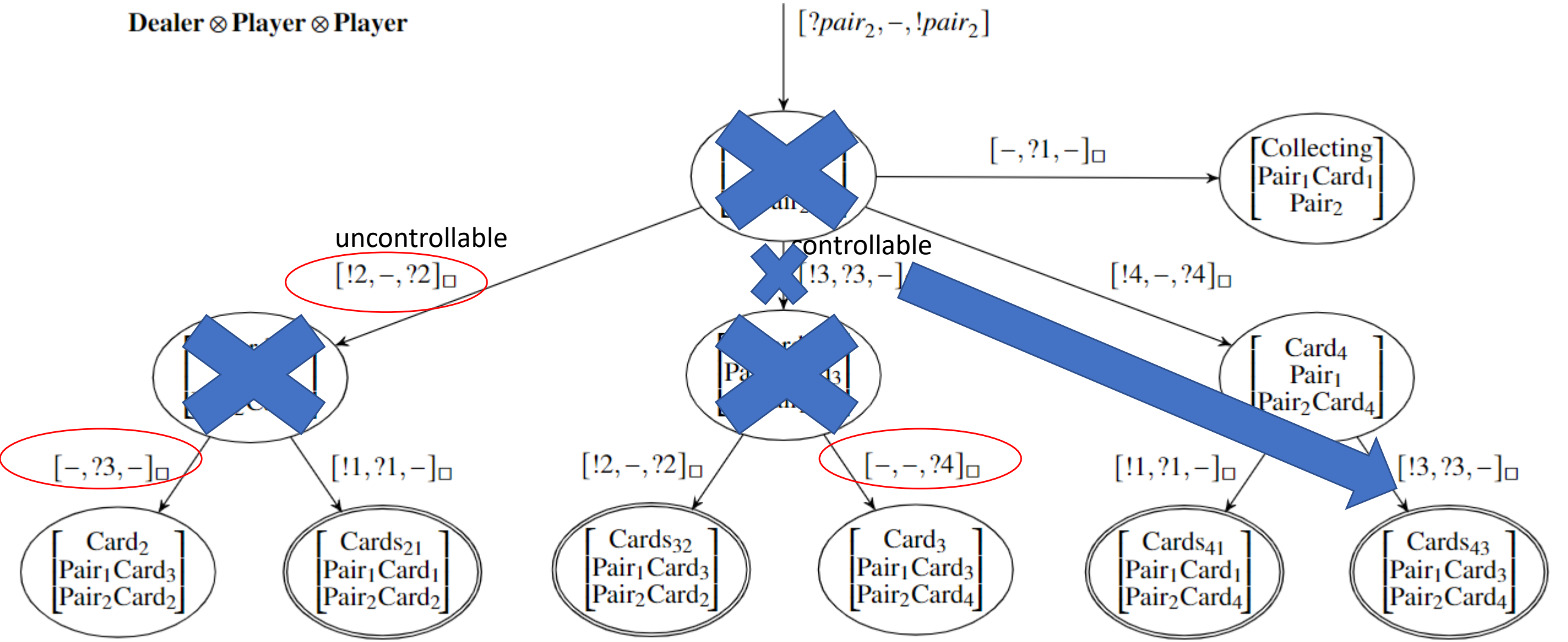

Dealer ⊗ Player ⊗ Player
[?pair2, −, !pair2]
[Collecting, Pair1, Pair2]
[−, ?1, −]□
[Collecting, Pair1 Card1, Pair2]
[!2, −, ?2]□
[!3, ?3, −]□
[!4, −, ?4]□
[Card3, Pair1 Card3, Pair2]
[Card4, Pair1, Pair2 Card4]
[−, ?3, −]□
[!1, ?1, −]□
[!2, −, ?2]□
[−, −, ?4]□
[!1, ?1, −]□
[!3, ?3, −]□
[Card2, Pair1 Card3, Pair2 Card2]
[Cards21, Pair1 Card1, Pair2 Card2]
[Cards32, Pair1 Card3, Pair2 Card2]
[Card3, Pair1 Card3, Pair2 Card4]
[Cards41, Pair1 Card1, Pair2 Card4]
[Cards43, Pair1 Card3, Pair2 Card4]

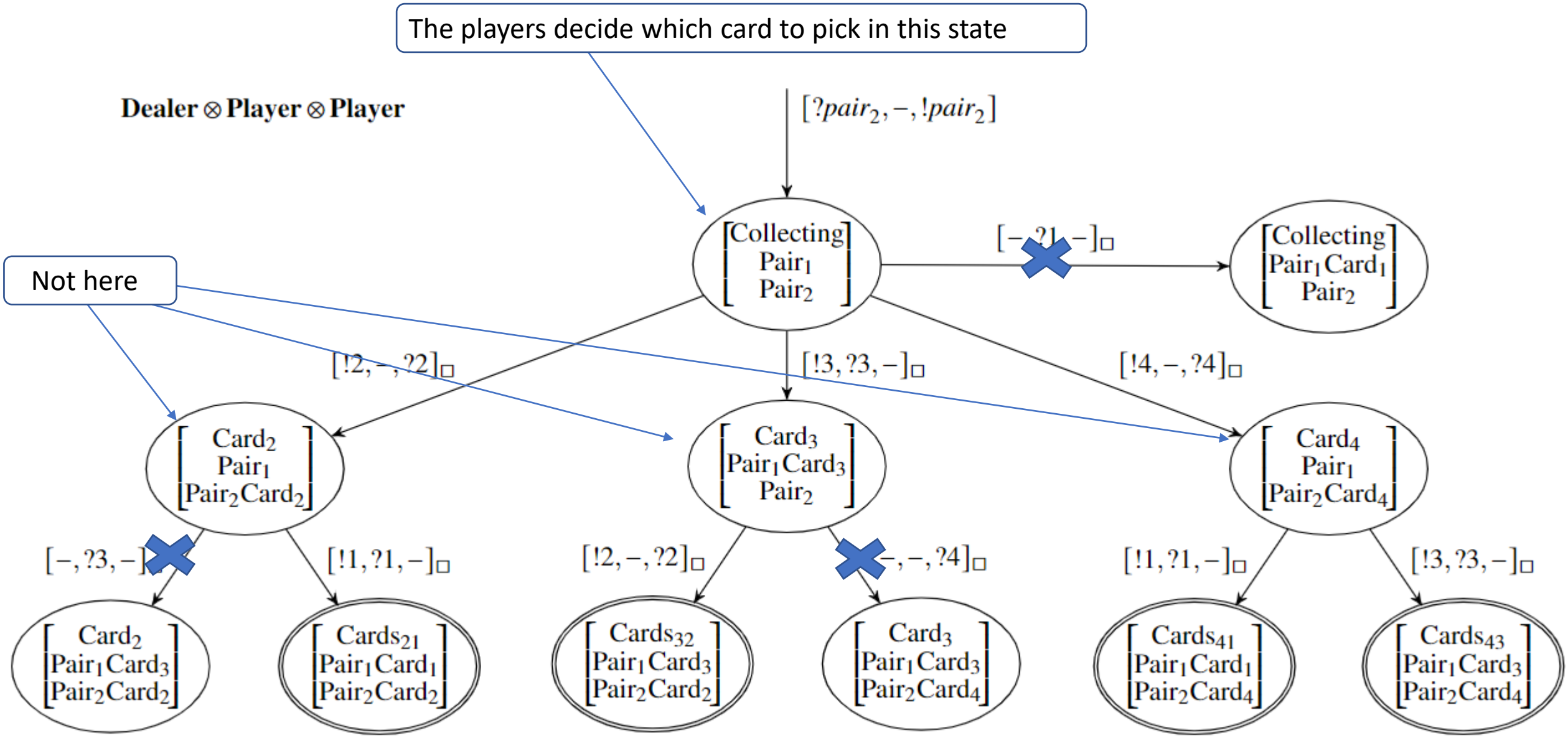

Dealer ⊗ Player ⊗ Player
[?pair2, −, !pair2]
[Collecting, Pair1, Pair2]
[−, ?1, −]□
[Collecting, Pair1 Card1, Pair2]
controllable
[!2, −, ?2]□
[!3, ?3, −]□
[!4, −, ?4]□
[Card3, Pair1 Card3, Pair2]
[Card4, Pair1, Pair2 Card4]
[−, ?3, −]□
[!1, ?1, −]□
[!2, −, ?2]□
[−, −, ?4]□
[!1, ?1, −]□
[!3, ?3, −]□
[Card2, Pair1 Card3, Pair2 Card2]
[Cards21, Pair1 Card1, Pair2 Card2]
[Cards32, Pair1 Card3, Pair2 Card2]
[Card3, Pair1 Card3, Pair2 Card4]
[Cards41, Pair1 Card1, Pair2 Card4]
[Cards43, Pair1 Card3, Pair2 Card4]


Dealer ⊗ Player ⊗ Player
[?pair2, −, !pair2]
[Collecting, Pair1, Pair2]
[−, ?1, −]□
[Collecting, Pair1 Card1, Pair2]
controllable
[!2, −, ?2]□
[!3, ?3, −]□
[!4, −, ?4]□
[Card3, Pair1 Card3, Pair2]
[Card4, Pair1, Pair2 Card4]
[−, ?3, −]□
[!1, ?1, −]□
[!2, −, ?2]□
[−, −, ?4]□
[!1, ?1, −]□
[!3, ?3, −]□
[Card2, Pair1 Card3, Pair2 Card2]
[Cards21, Pair1 Card1, Pair2 Card2]
[Cards32, Pair1 Card3, Pair2 Card2]
[Card3, Pair1 Card3, Pair2 Card4]
[Cards41, Pair1 Card1, Pair2 Card4]
[Cards43, Pair1 Card3, Pair2 Card4]


Dealer ⊗ Player ⊗ Player
[?pair2, −, !pair2]
[Collecting, Pair1, Pair2]
[−, ?1, −]□
[Collecting, Pair1 Card1, Pair2]
controllable
[!2, −, ?2]□
controllable
[!3, ?3, −]
[!4, −, ?4]□
[Card4, Pair1, Pair2 Card4]
[−, ?3, −]□
[!1, ?1, −]□
[!2, −, ?2]□
[−, −, ?4]□
[!1, ?1, −]□
[!3, ?3, −]□
[Card2, Pair1 Card3, Pair2 Card2]
[Cards21, Pair1 Card1, Pair2 Card2]
[Cards32, Pair1 Card3, Pair2 Card2]
[Card3, Pair1 Card3, Pair2 Card4]
[Cards41, Pair1 Card1, Pair2 Card4]
[Cards43, Pair1 Card3, Pair2 Card4]


Dealer ⊗ Player ⊗ Player
[?pair2, −, !pair2]
[Collecting, Pair1, Pair2]
[−, ?1, −]□
[Collecting, Pair1 Card1, Pair2]
[!2, −, ?2]□
[!3, ?3, −]
[!4, −, ?4]□
controllable
[Card4, Pair1, Pair2 Card4]
[−, ?3, −]□
[!1, ?1, −]□
[!2, −, ?2]□
[−, −, ?4]□
[!1, ?1, −]□
[!3, ?3, −]□
[Card2, Pair1 Card3, Pair2 Card2]
[Cards21, Pair1 Card1, Pair2 Card2]
[Cards32, Pair1 Card3, Pair2 Card2]
[Card3, Pair1 Card3, Pair2 Card4]
[Cards41, Pair1 Card1, Pair2 Card4]
[Cards43, Pair1 Card3, Pair2 Card4]


Dealer ⊗ Player ⊗ Player
[?pair2, −, !pair2]
[−, ?1, −]□
[Collecting Pair1 Card1 Pair2]
uncontrollable
[!2, −, ?2]□
controllable
[!3, ?3, −]
[!4, −, ?4]□
[Card4 Pair1 Pair2 Card4]
[−, ?3, −]□
[!1, ?1, −]□
[!2, −, ?2]□
[−, −, ?4]□
[!1, ?1, −]□
[!3, ?3, −]□
[Card2 Pair1 Card3 Pair2 Card2]
[Cards21 Pair1 Card1 Pair2 Card2]
[Cards32 Pair1 Card3 Pair2 Card2]
[Card3 Pair1 Card3 Pair2 Card4]
[Cards41 Pair1 Card1 Pair2 Card4]
[Cards43 Pair1 Card3 Pair2 Card4]


The players decide which card to pick in this state
Dealer ⊗ Player ⊗ Player
$[?pair_2, -, !pair_2]$
Not here
[Collecting, Pair1, Pair2]
$[-, ?1, -]_\square$
[Collecting, Pair1 Card1, Pair2]
$[!2, -, ?2]_\square$
$[!3, ?3, -]_\square$
$[!4, -, ?4]_\square$
[Card2, Pair1, Pair2 Card2]
[Card3, Pair1 Card3, Pair2]
[Card4, Pair1, Pair2 Card4]
$[-, ?3, -]_\square$
$[!1, ?1, -]_\square$
$[!2, -, ?2]_\square$
$[-, -, ?4]_\square$
$[!1, ?1, -]_\square$
$[!3, ?3, -]_\square$
[Card2, Pair1 Card3, Pair2 Card2]
[Cards21, Pair1 Card1, Pair2 Card2]
[Cards32, Pair1 Card3, Pair2 Card2]
[Card3, Pair1 Card3, Pair2 Card4]
[Cards41, Pair1 Card1, Pair2 Card4]
[Cards43, Pair1 Card3, Pair2 Card4]

Conclusion

- We have discussed a refinement of the notion of semi-controllability and open challenges in the synthesis of orchestrations in contract automata
- The paper also indicates further challenges and a research roadmap to tackle these challenges effectively

Conclusion

- We have discussed a refinement of the notion of semi-controllability and open challenges in the synthesis of orchestrations in contract automata
- The paper also indicates further challenges and a research roadmap to tackle these challenges effectively
- **Thanks for your attention!**