Models and Theory of Computation (MTC)
EPFL

Dirk Beyer, Jasmin Fisher, Nir Piterman

Simon Kramer: Logic for cryptography
Marc Schaub: Models for biological systems
Vasu Singh: Software interface derivation
Gregory Theoduloz: Combining model checking and program analysis

Arindam Chakrabarti: Web service interfaces
Krishnendu Chatterjee: Stochastic games
Slobodan Matic: Time-triggered programming
Vinayak Prabhu: Robust hybrid systems

Model Checking:
From Graphs to Games

Tom Henzinger
EPFL

Graph Models of Systems

vertices = states

edges = transitions

paths = behaviors

Game Models of Systems

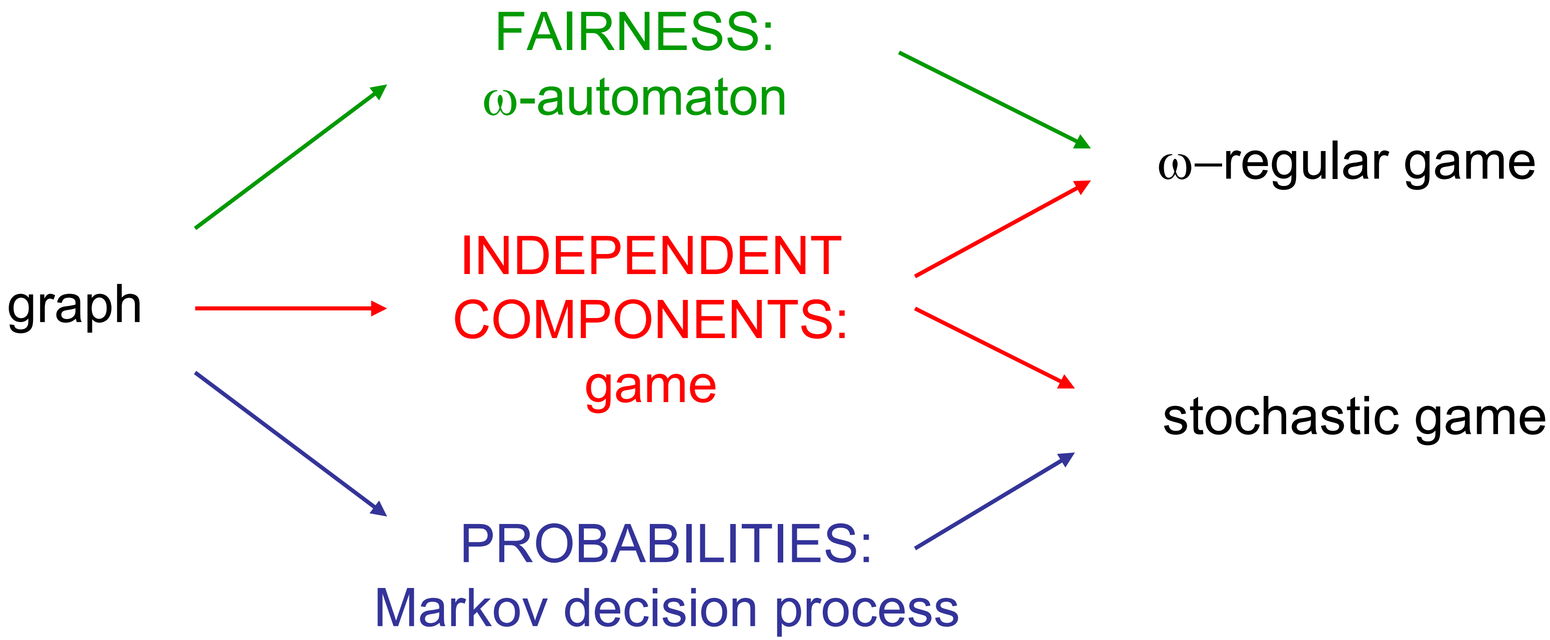
vertices = states

edges = transitions

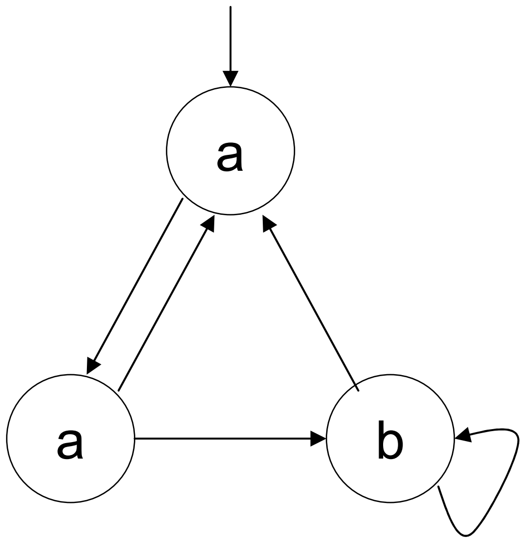
paths = behaviors

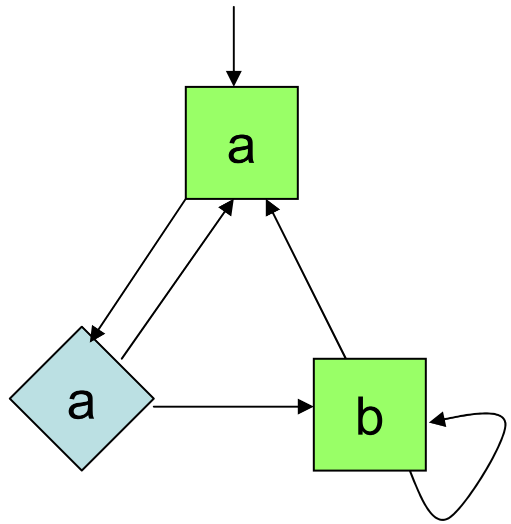
players = components

Game Models of Systems



Graphs vs. Games





Game Models enable

-synthesis [Church, Rabin, Ramadge/Wonham, Pnueli/Rosner et al.]

-receptiveness [Dill, Abadi/Lamport]

-semantics of interaction [Abramsky]

-reasoning about adversarial behavior

-interface-based design

-modular reasoning [Kupferman/Vardi et al.]

-early error detection [deAlfaro/H/Mang]

-model-based testing [Gurevich et al.]

-scheduling [Sifakis et al.]

-reasoning about security [Raskin et al.]

-etc.

Game Models

Always about open systems:

-players = processes / components / agents

-input vs. output

-demonic vs. angelic nondeterminism

Game Models

Always about open systems:

-players = processes / components / agents

-input vs. output

-demonic vs. angelic nondeterminism

1. Output games: input demonic (adversarial)
2. Input games: output demonic

Output Games

P1:

```
init  x := 0
loop
    choice
        |  x := x+1 mod 2
        |  x := 0
    end choice
end loop
```

P2:

```
init  y := 0
loop
    choice
        |  y := x
        |  y := x+1 mod 2
    end choice
end loop
```

S1: $\Box\,(x \geq y)$

S2: $\Box$ even(y)

Graph Questions

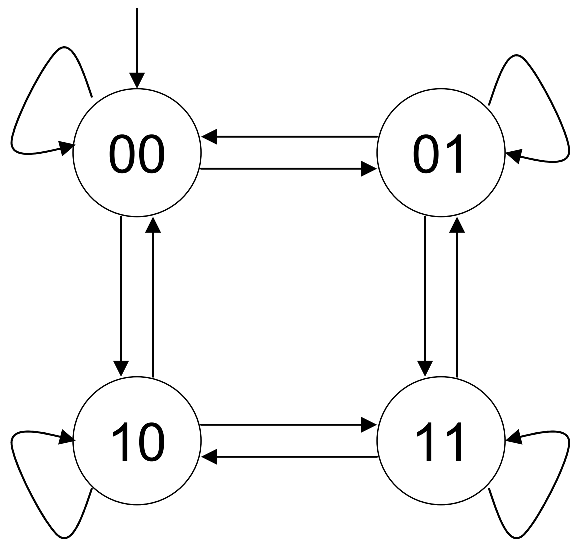
$\forall \Box (x \geq y)$

$\exists \Box (x \geq y)$

Graph Questions

✗ $\forall \Box (x \geq y)$

✓ $\exists \Box (x \geq y)$



Zero-Sum Game Questions

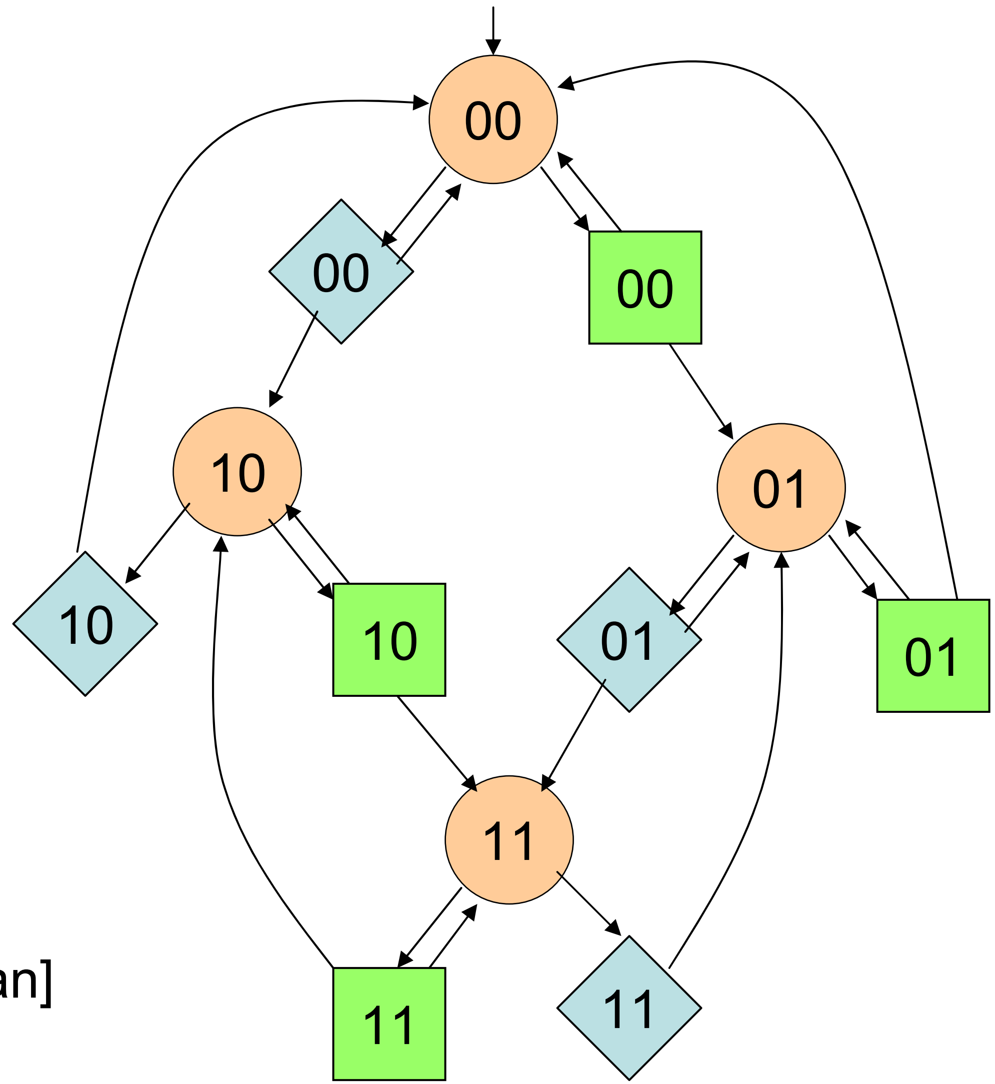
$\langle\langle P1 \rangle\rangle \,\square\, (x \geq y)$

$\langle\langle P2 \rangle\rangle \,\square\, \text{even}(y)$

Zero-Sum Game Questions

✗ $\langle\langle P1 \rangle\rangle \, \Box \, (x \geq y)$

✓ $\langle\langle P2 \rangle\rangle \, \Box \, even(y)$



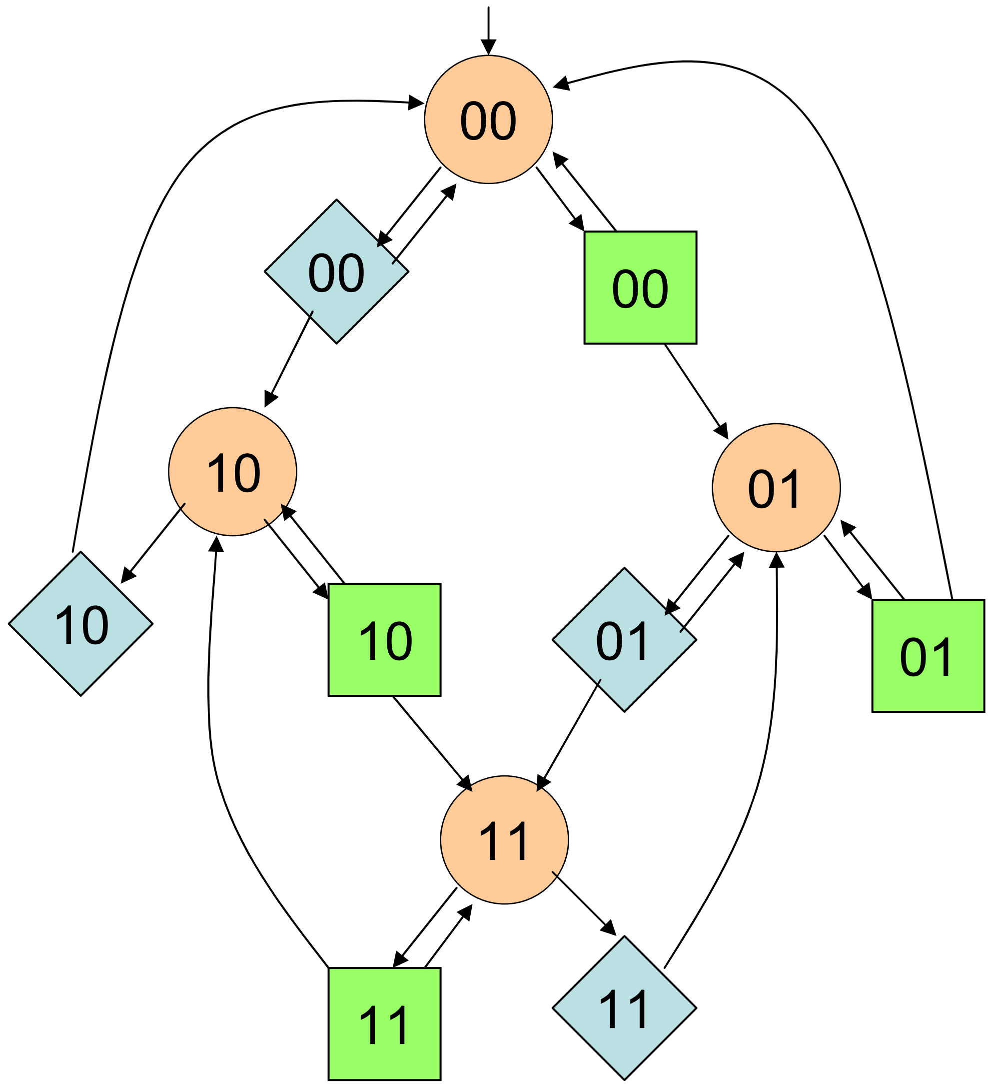
ATL [Alur/H/Kupferman]

Nonzero-Sum Game Questions

$\langle\langle P1 \rangle\rangle \; \Box \; (x \geq y)$

$\otimes$

$\langle\langle P2 \rangle\rangle \; \Box \; \mathrm{even}(y)$

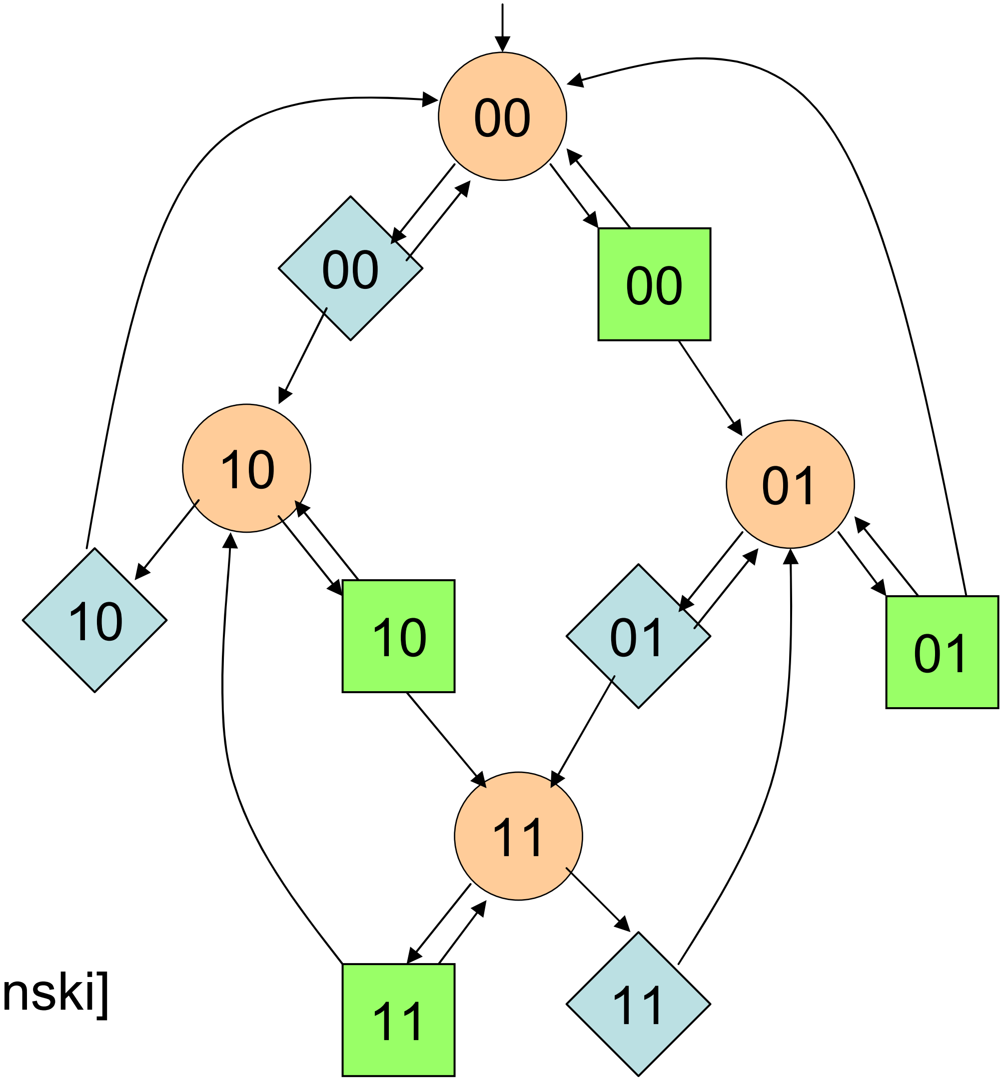


Nonzero-Sum Game Questions

$\langle\langle P1 \rangle\rangle \; \Box \; (x \geq y)$

✓ $\otimes$

$\langle\langle P2 \rangle\rangle \; \Box \; \text{even}(y)$



Secure equilibrium
[Chatterjee/H/Jurdzinski]

Classical Notion of Rationality

Nash equilibrium: none of the players gains by deviation.

(row, column)

3,1	1,0
3,2	4,2

Refined Notion of Rationality

Nash equilibrium: none of the players gains by deviation.

Secure equilibrium: none hurts the opponent by deviation.

(row, column)

3,1	1,0
3,2	4,2

Secure Equilibrium

- Natural notion of rationality for multi-component systems:
  - First, a component tries to meet its specification.
  - Second, a component may obstruct the other components.
- A secure equilibrium is a contract:

  if one player deviates to lower the other player's payoff,
  then her own payoff decreases as well, and vice versa.

Theorem

$\mathbf{W_{01}}$
$\langle\langle P2 \rangle\rangle\,(S2 \wedge \neg\, S1)$

$\mathbf{W_{00}}$

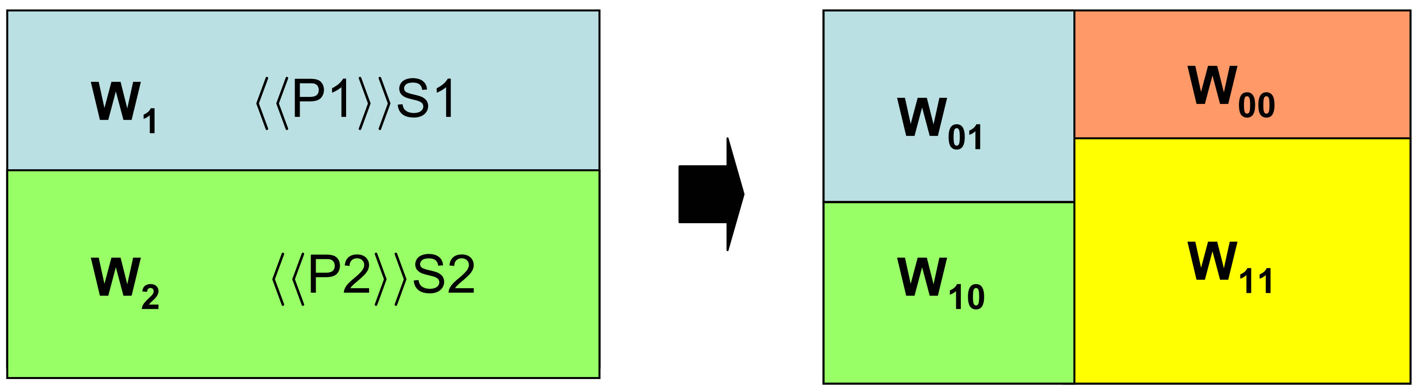
$\mathbf{W_{10}}$
$\langle\langle P1 \rangle\rangle\,(S1 \wedge \neg\, S2)$

$\mathbf{W_{11}}$
$\langle\langle P1 \rangle\rangle S1 \otimes \langle\langle P2 \rangle\rangle S2$

Generalization of Determinacy

Zero-sum games: S1 = ¬S2

Nonzero-sum games: S1, S2



Game Models

Always about open systems:

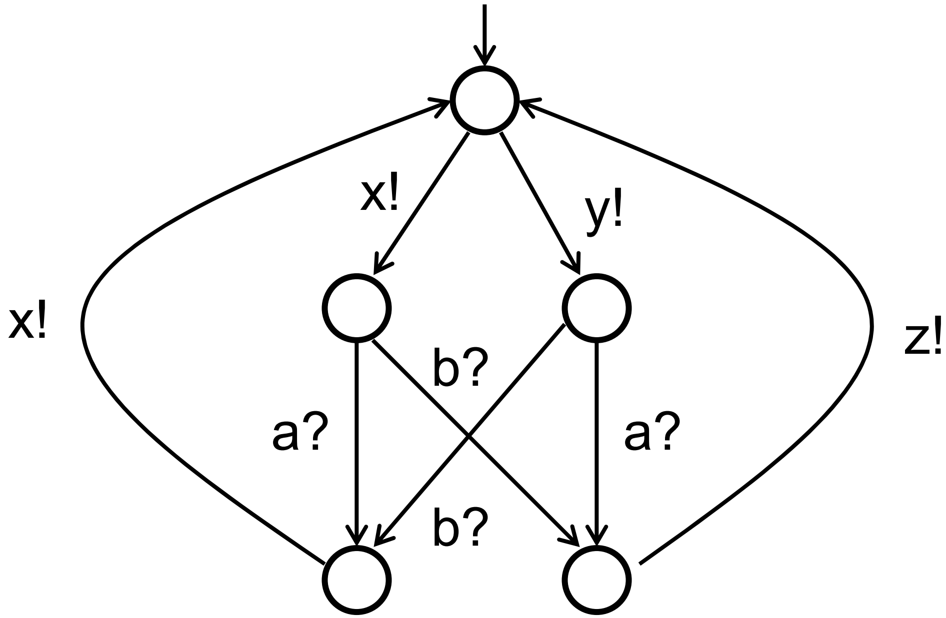
-players = processes / components / agents

-input vs. output

-demonic vs. angelic nondeterminism

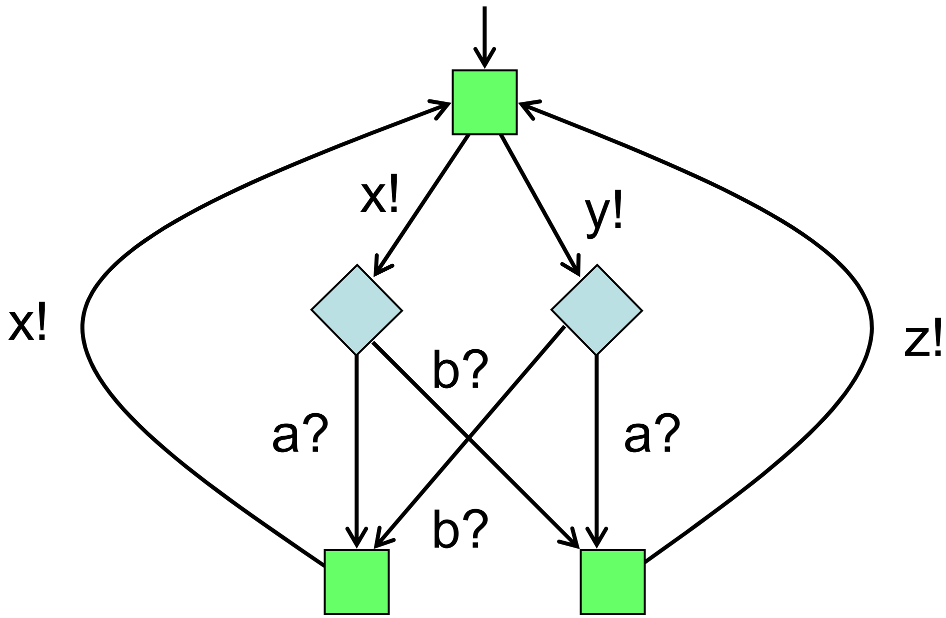
1. Output games: input demonic (adversarial)
2. Input games: output demonic

Input Games



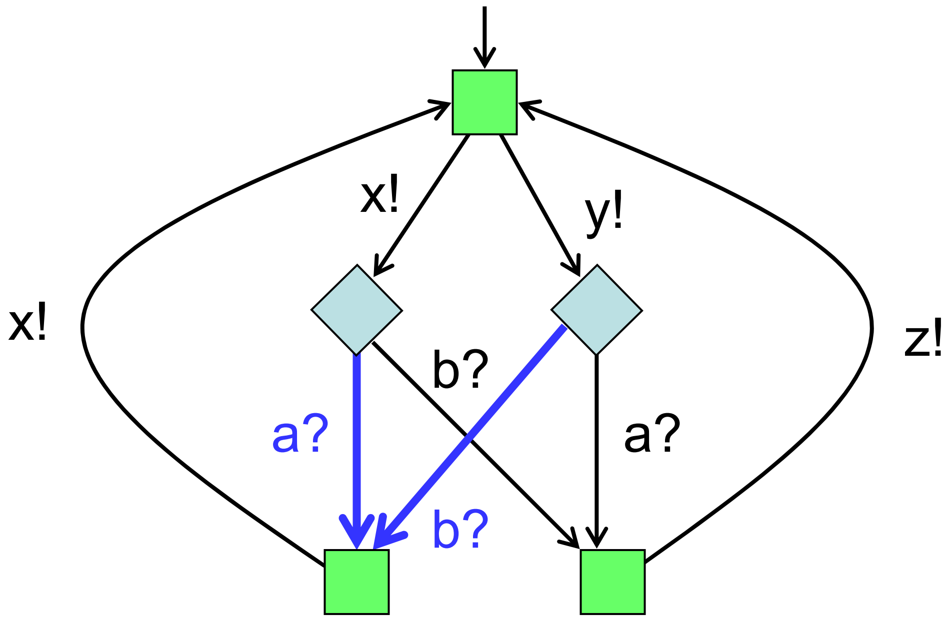
Control objective:
□ ¬ z

Input Games



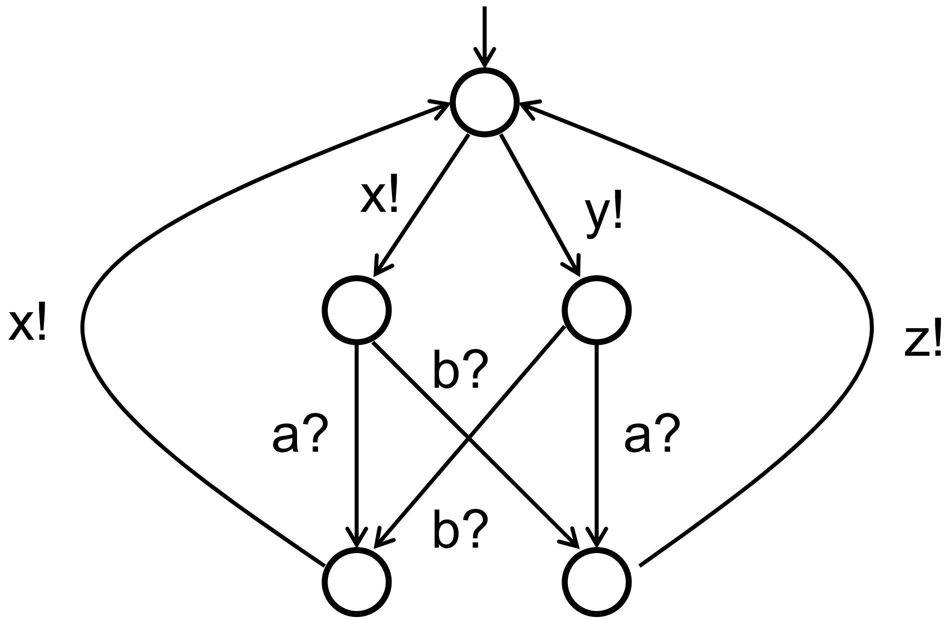
Control objective:
$\Box \neg z$

Input Games

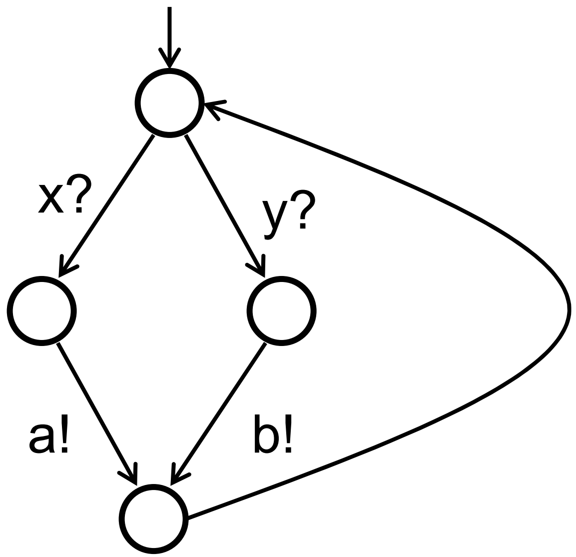


Control objective:
$\square \neg z$

Input Games

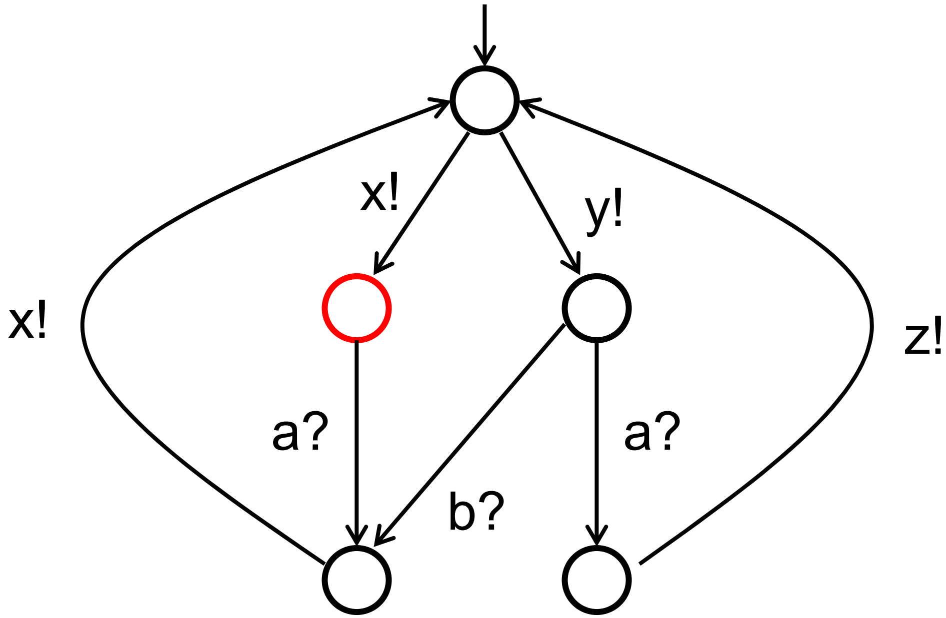


Controller:



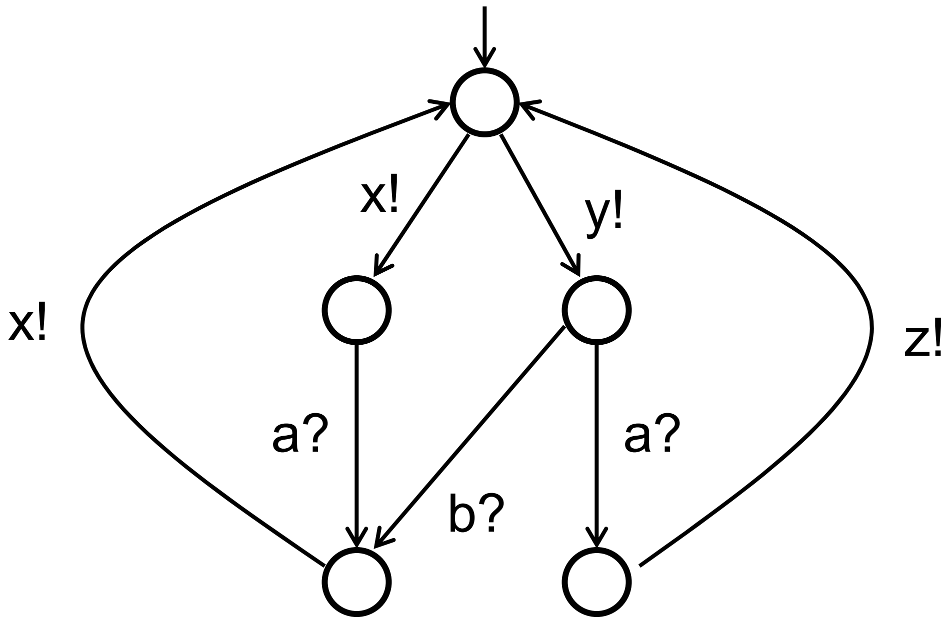
[Ramadge/Wonham et al.]

Input Games



Not input enabling

Input Games



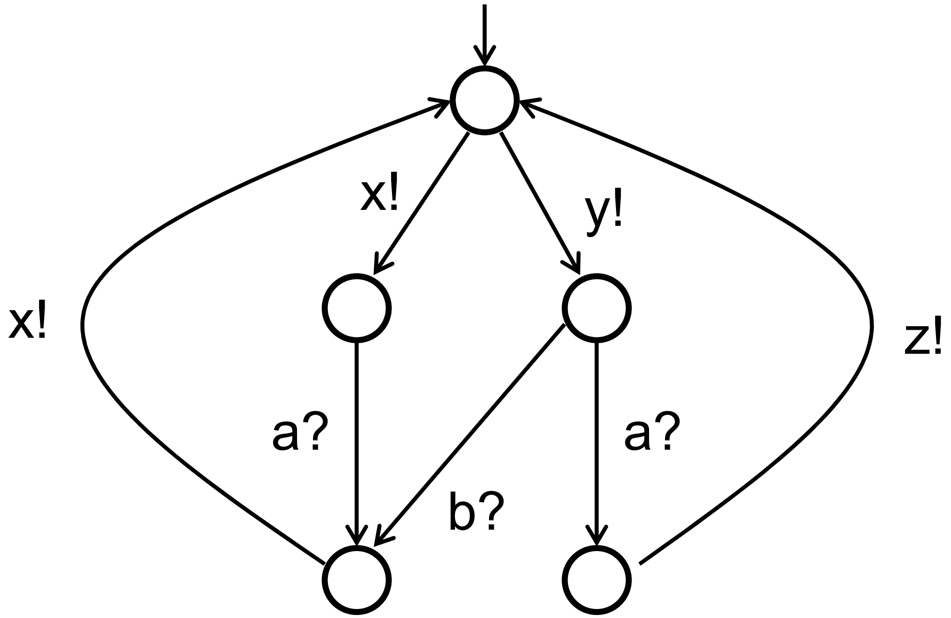
Environment avoids deadlock

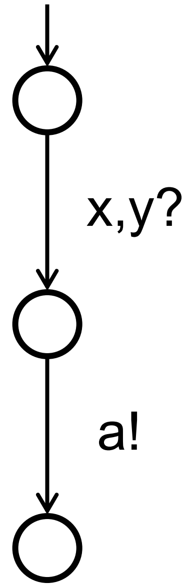
=

Input assumption

Interface automata [deAlfaro/H]

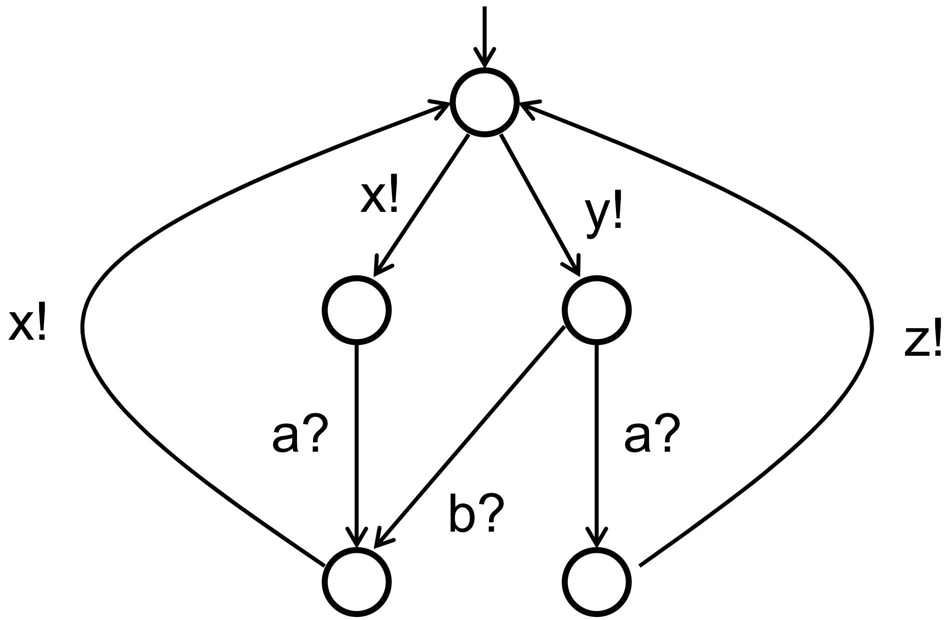
Input Games

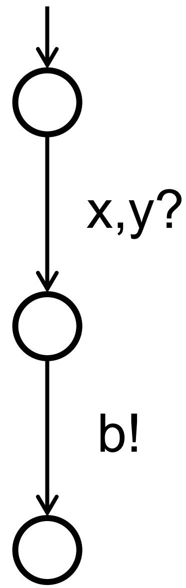




Legal environment

Input Games

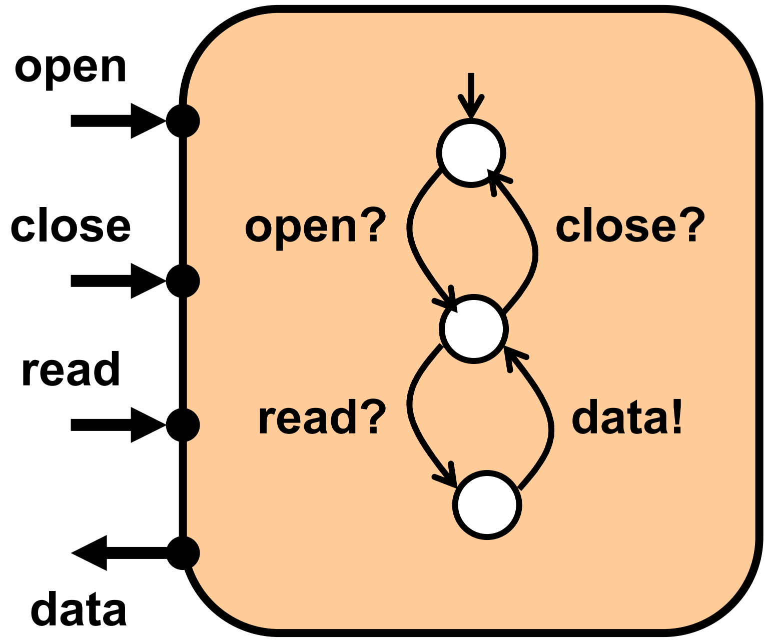




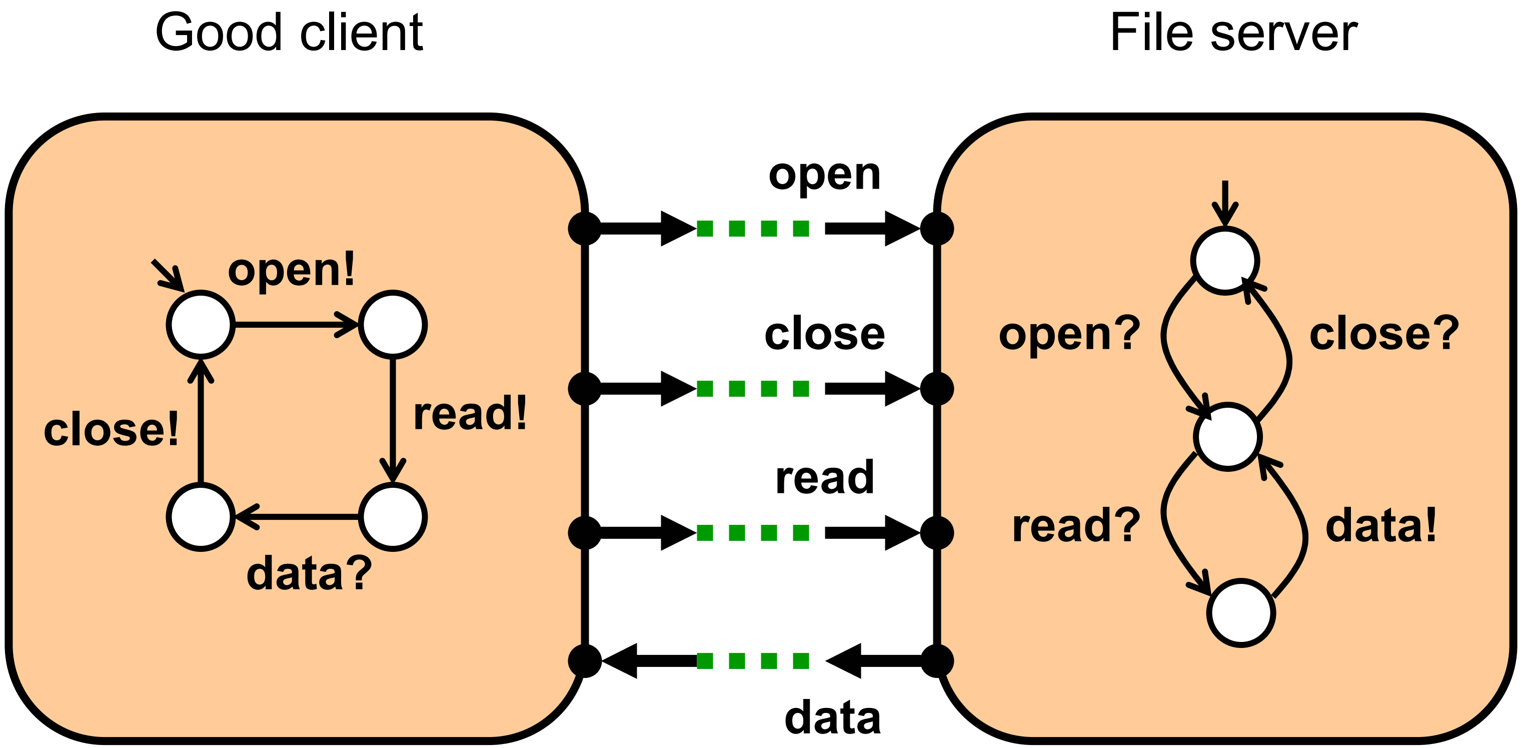
Illegal environment

Interface Compatibility

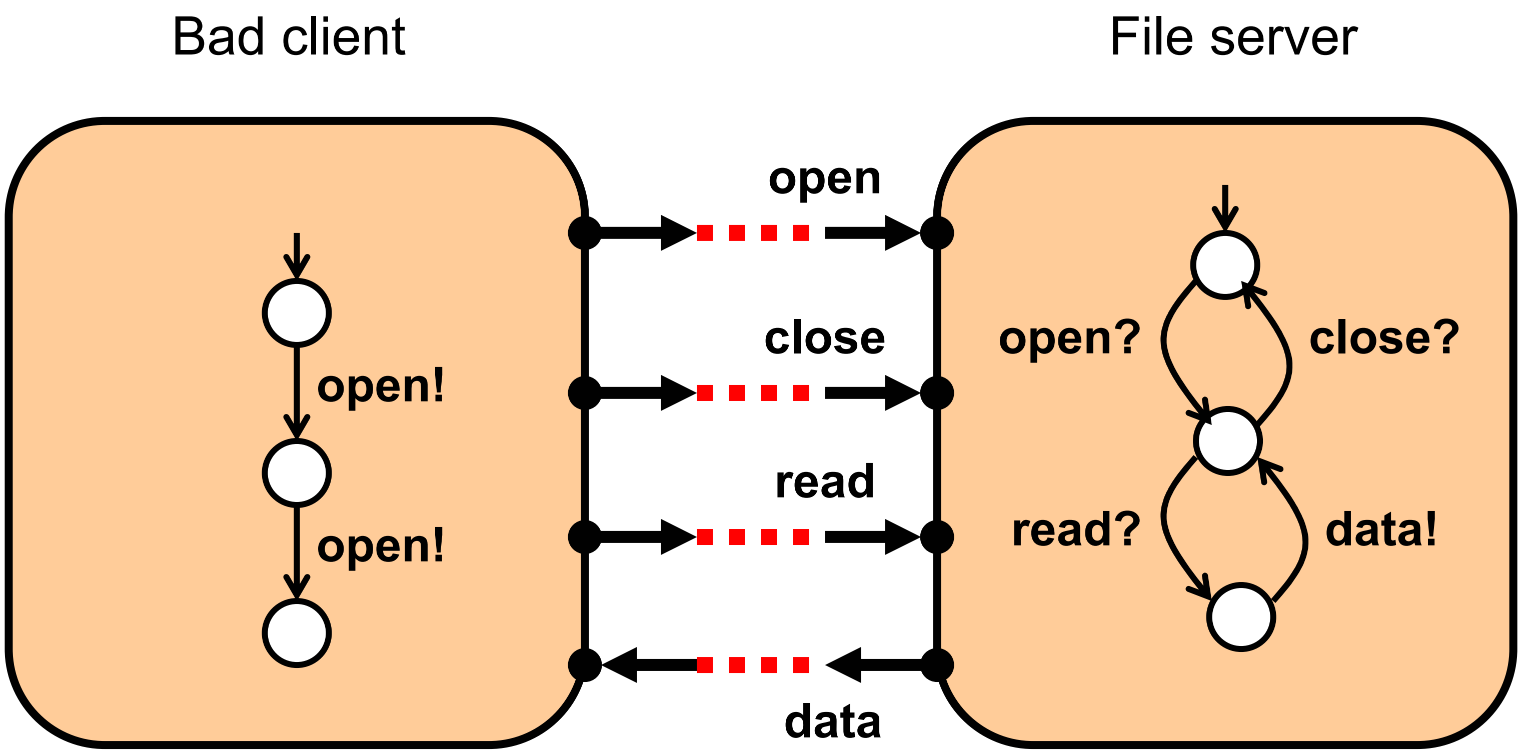
File server



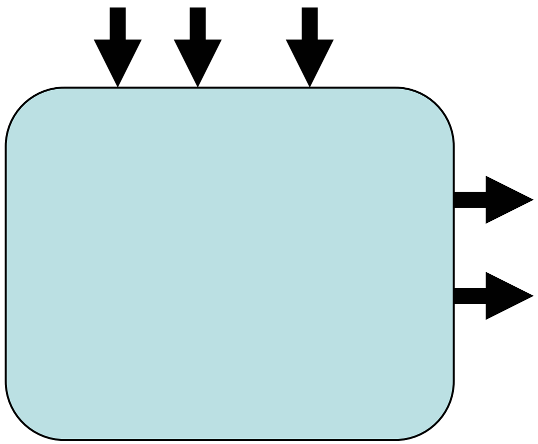
Interface Compatibility



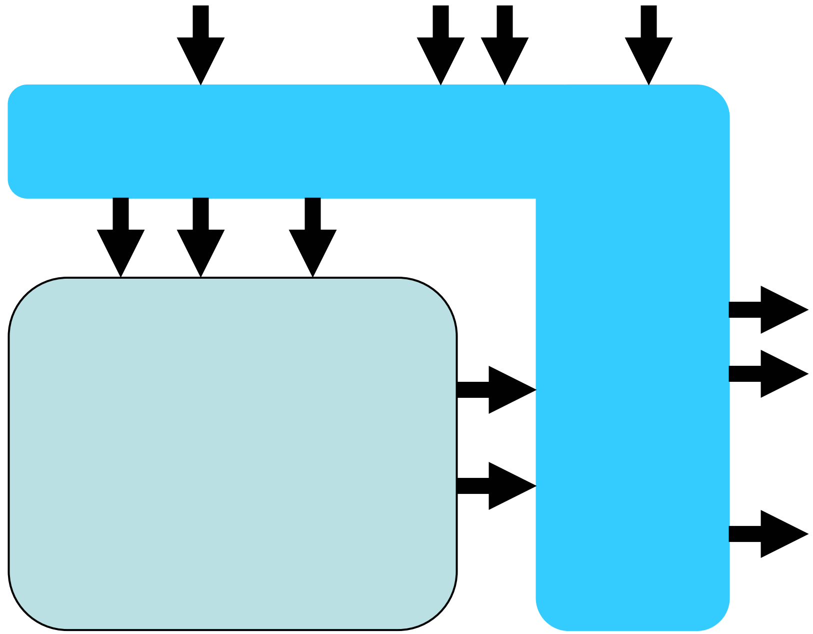
Interface Compatibility



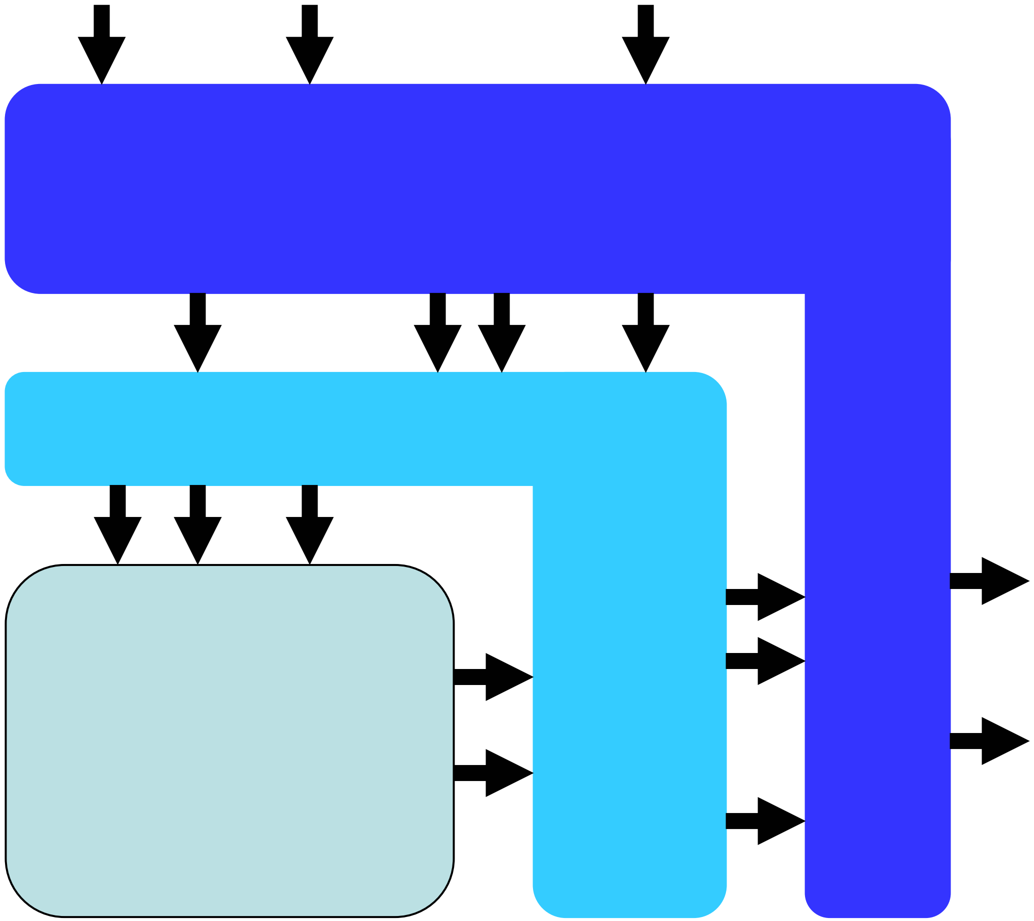
Incremental Design



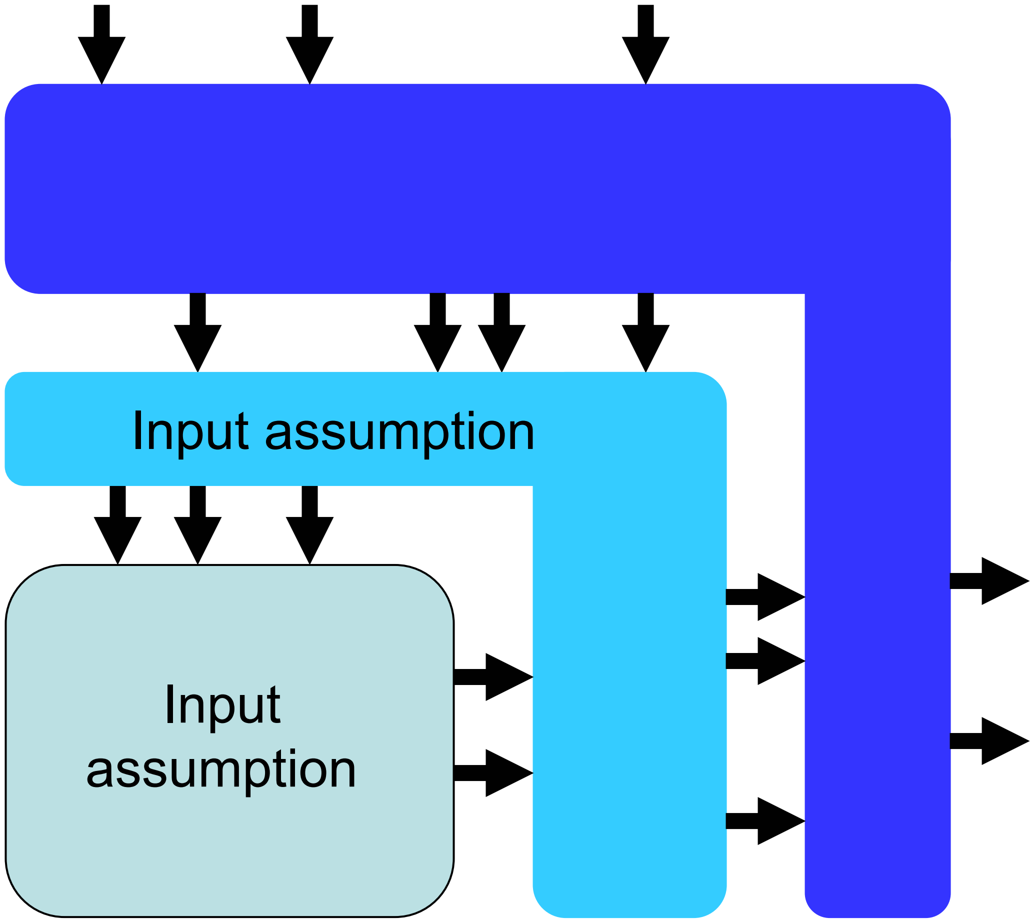
Incremental Design



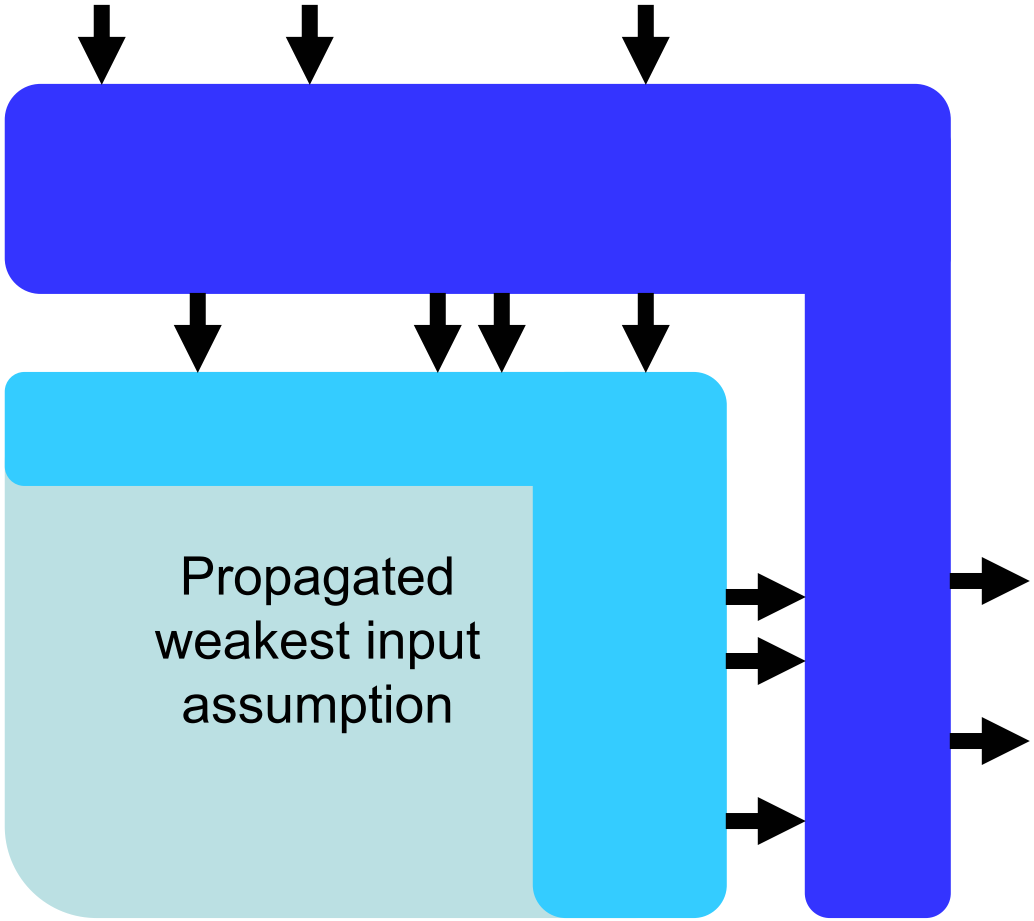
Incremental Design



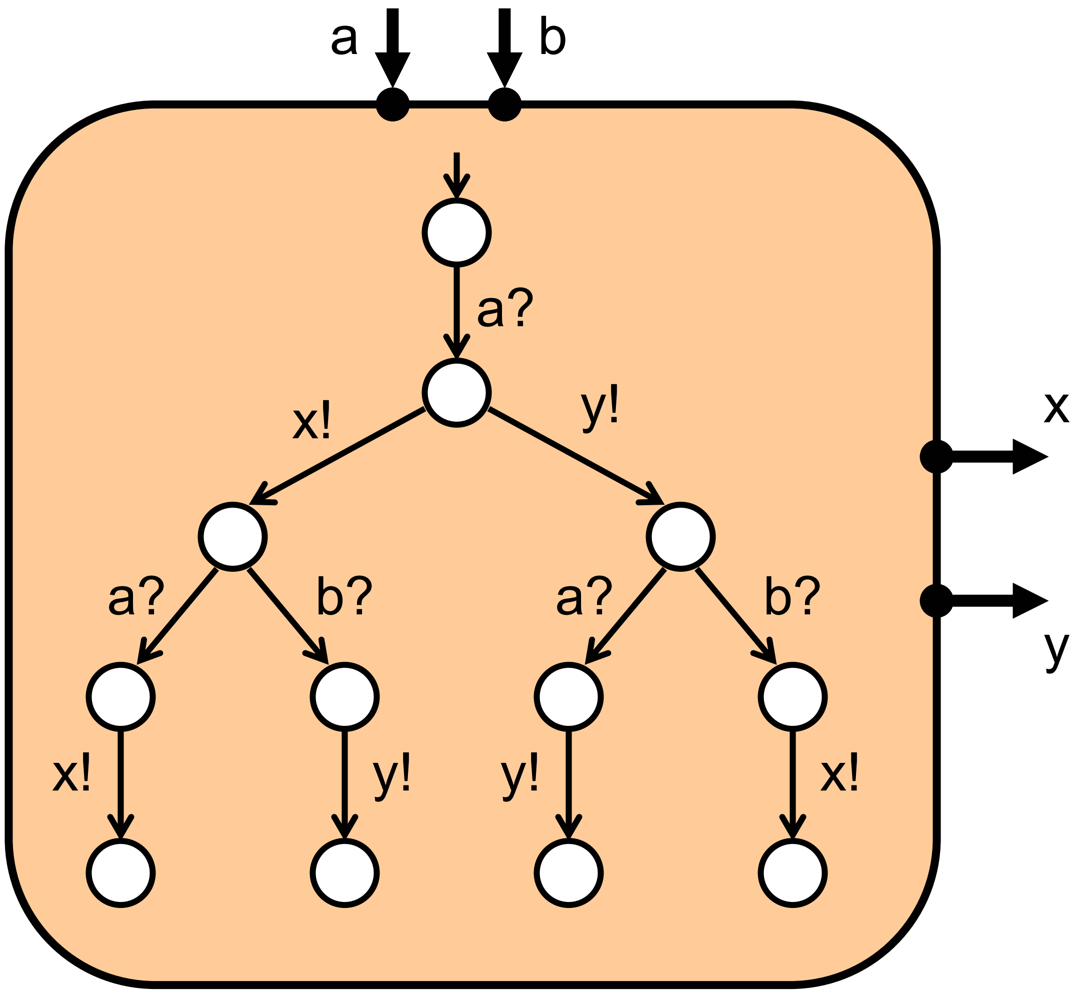
Incremental Design



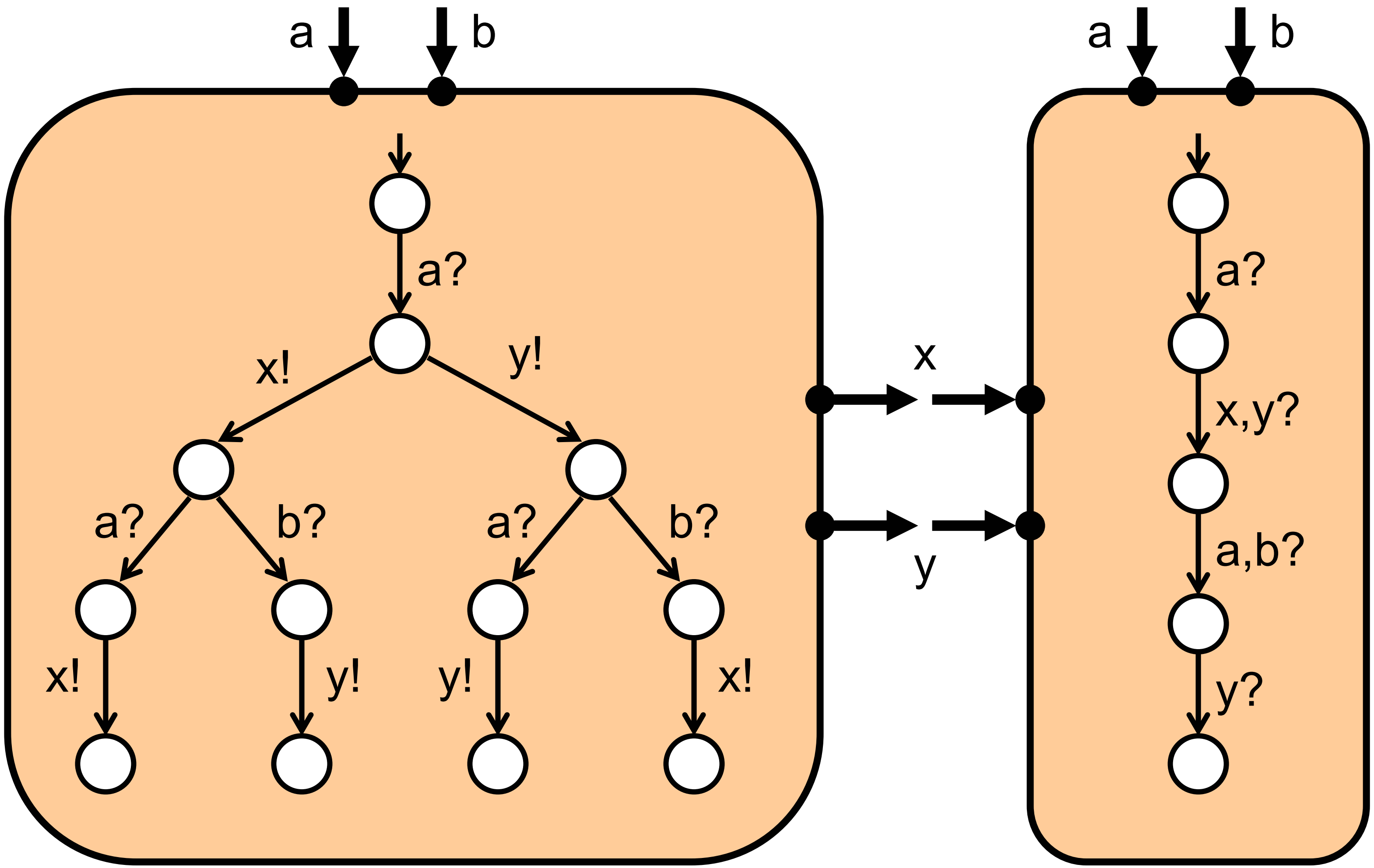
Incremental Design



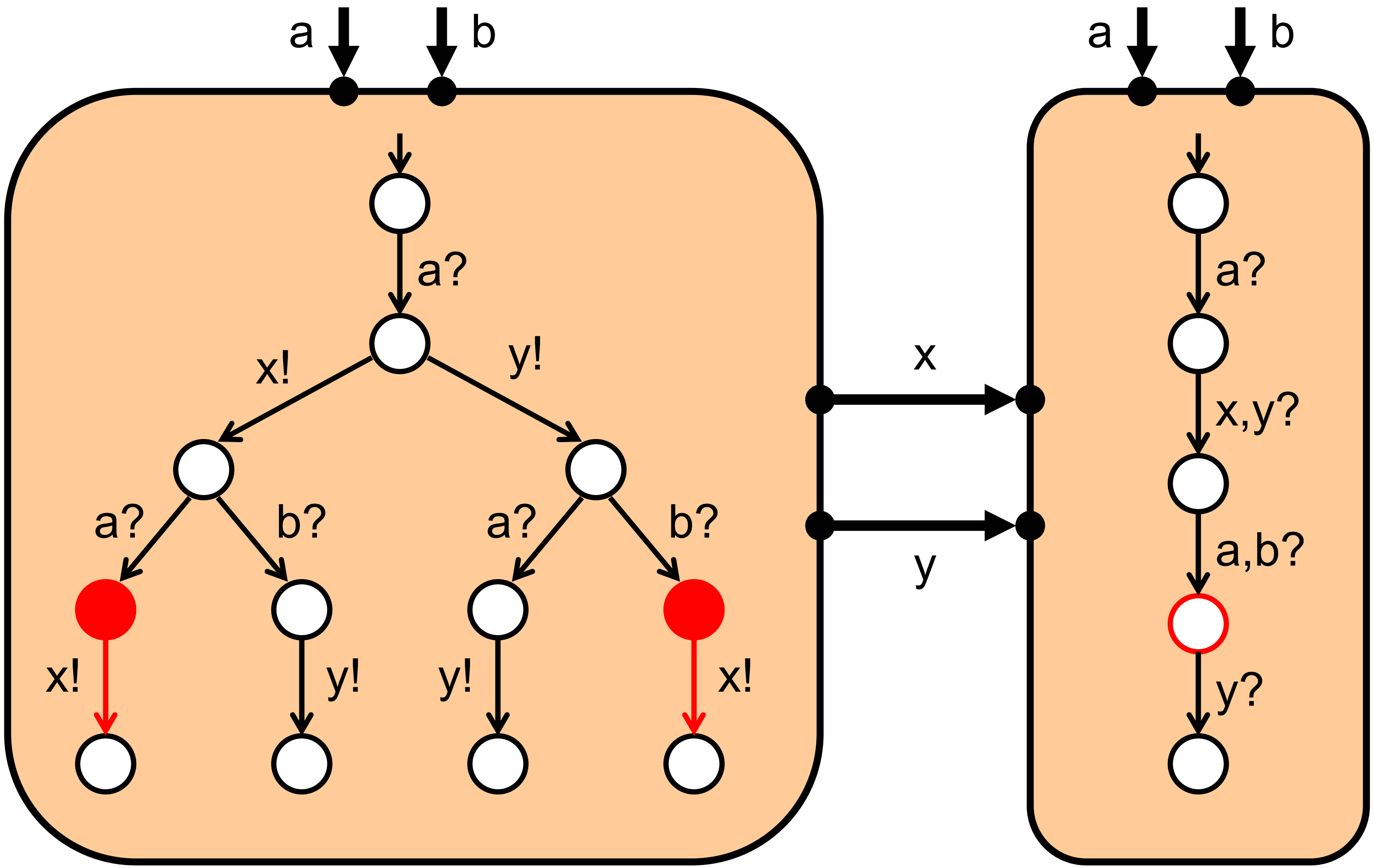
Input Assumption Propagation



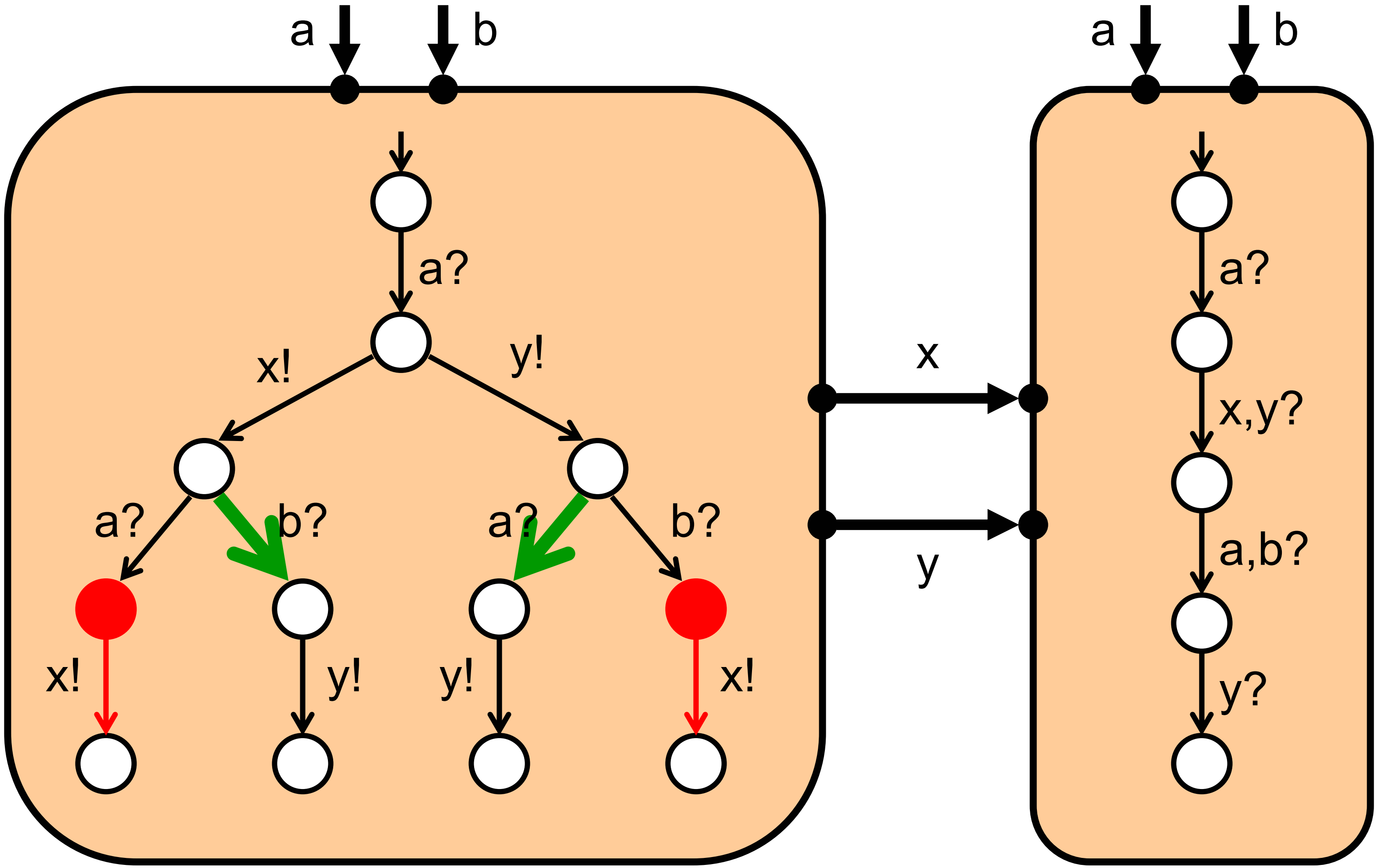
Input Assumption Propagation



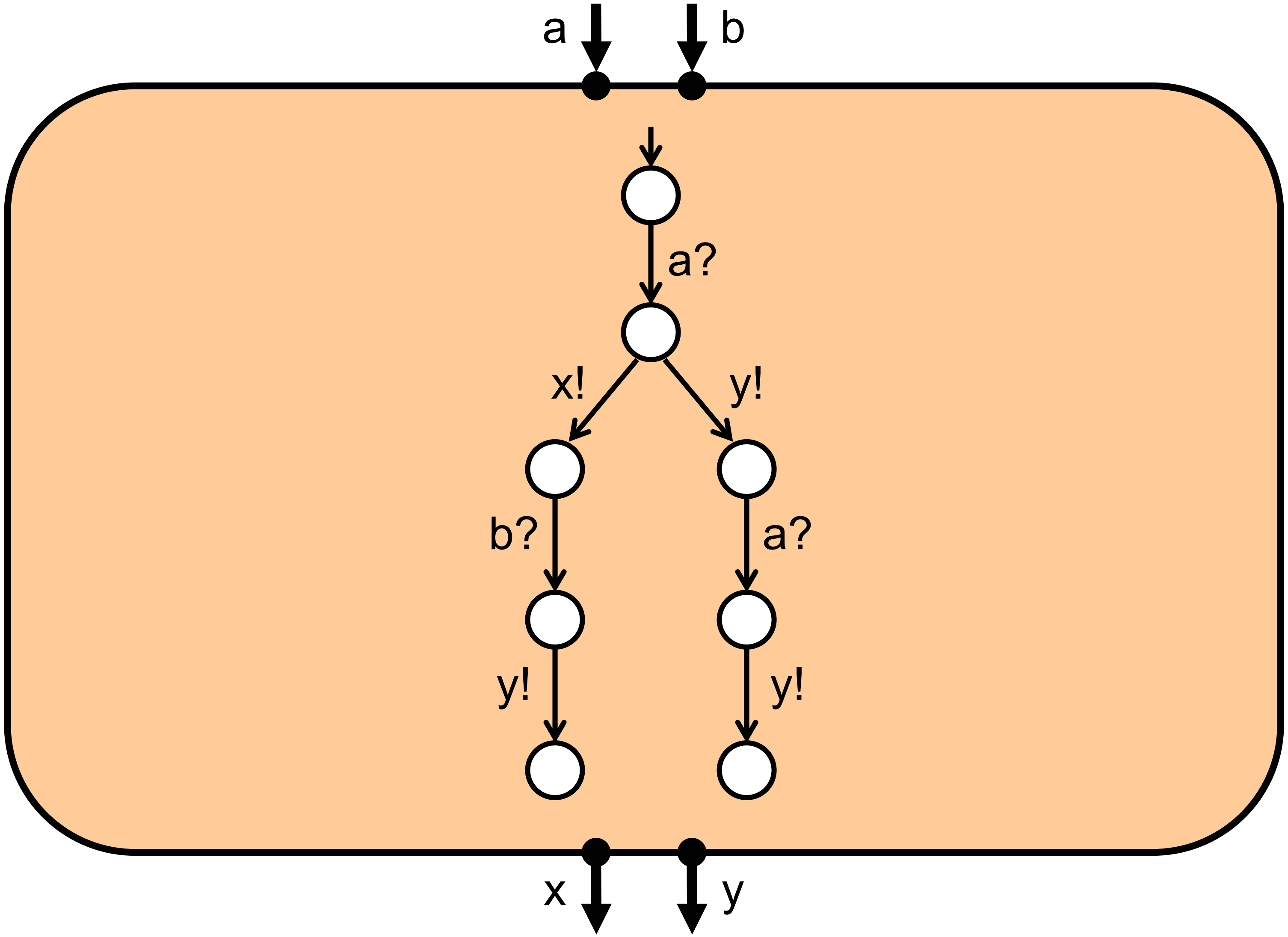
Input Assumption Propagation



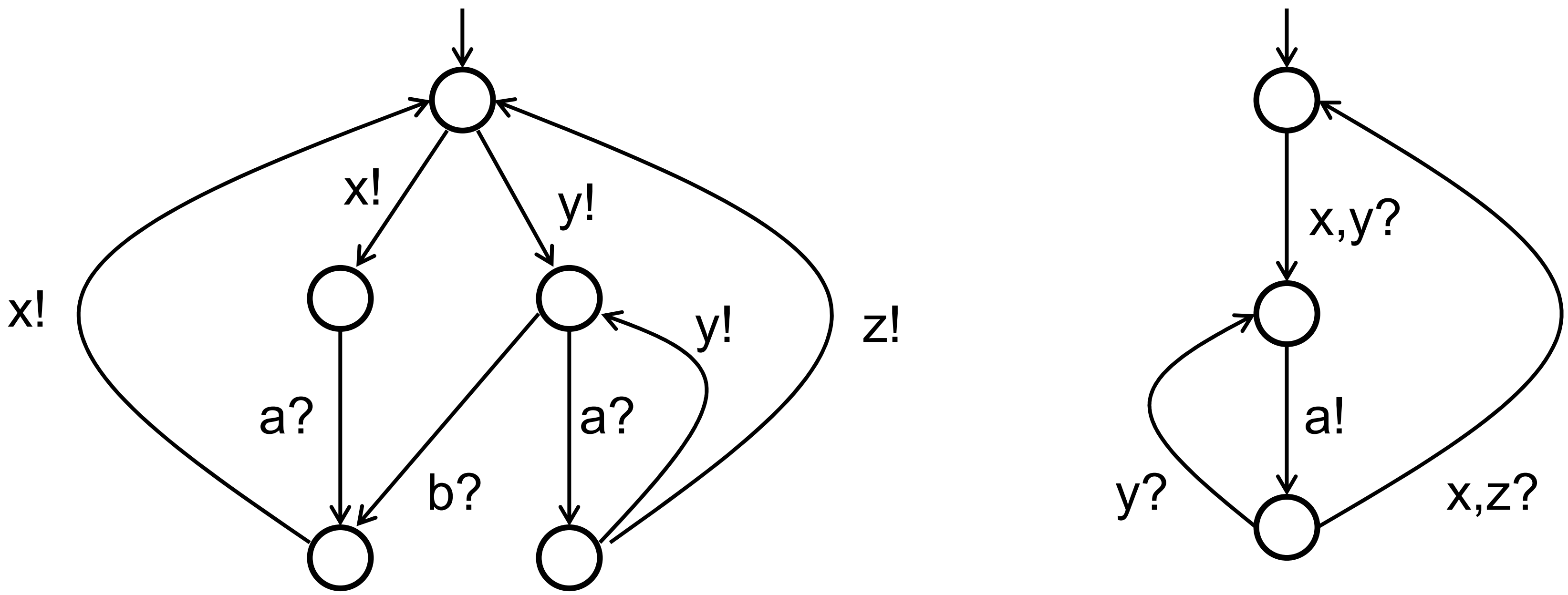
Two interfaces are compatible if they can be used together in ***some*** environment.



The Composite Interface

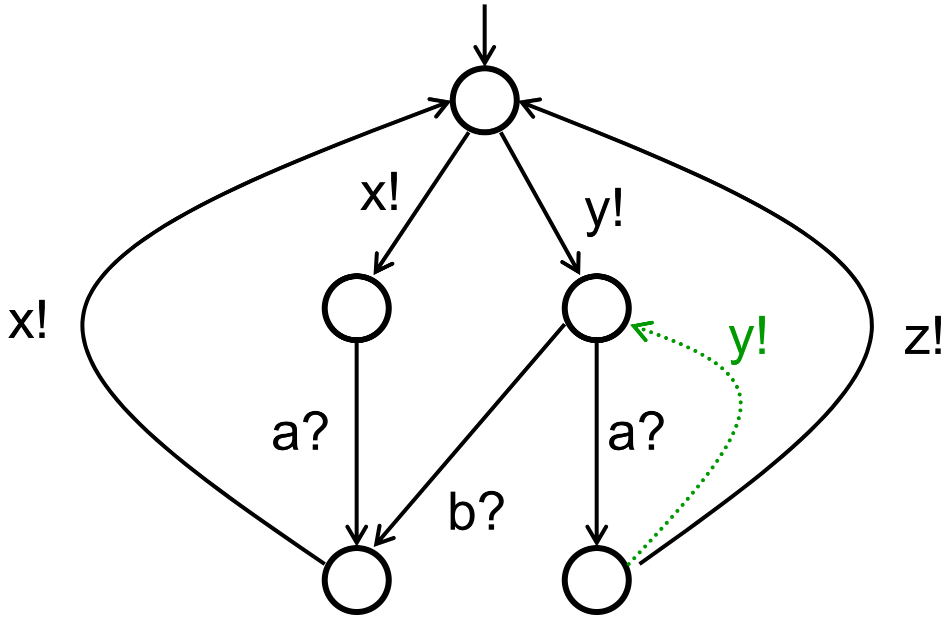


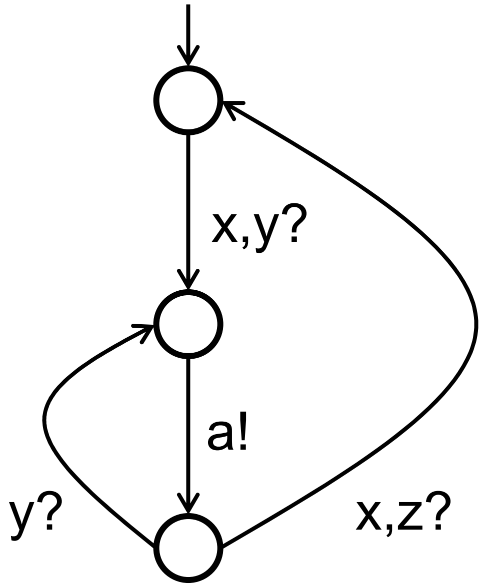
Refinement



Every legal environment should be a legal environment of the refined process.

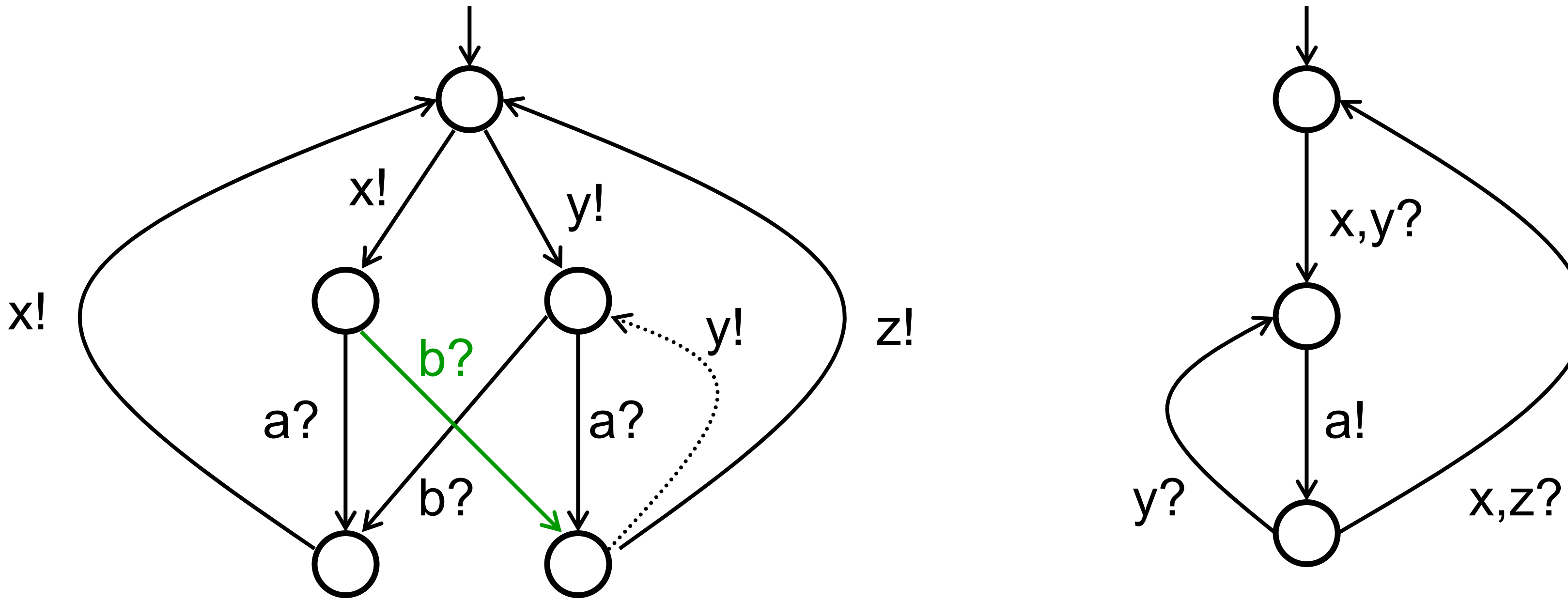
Refinement





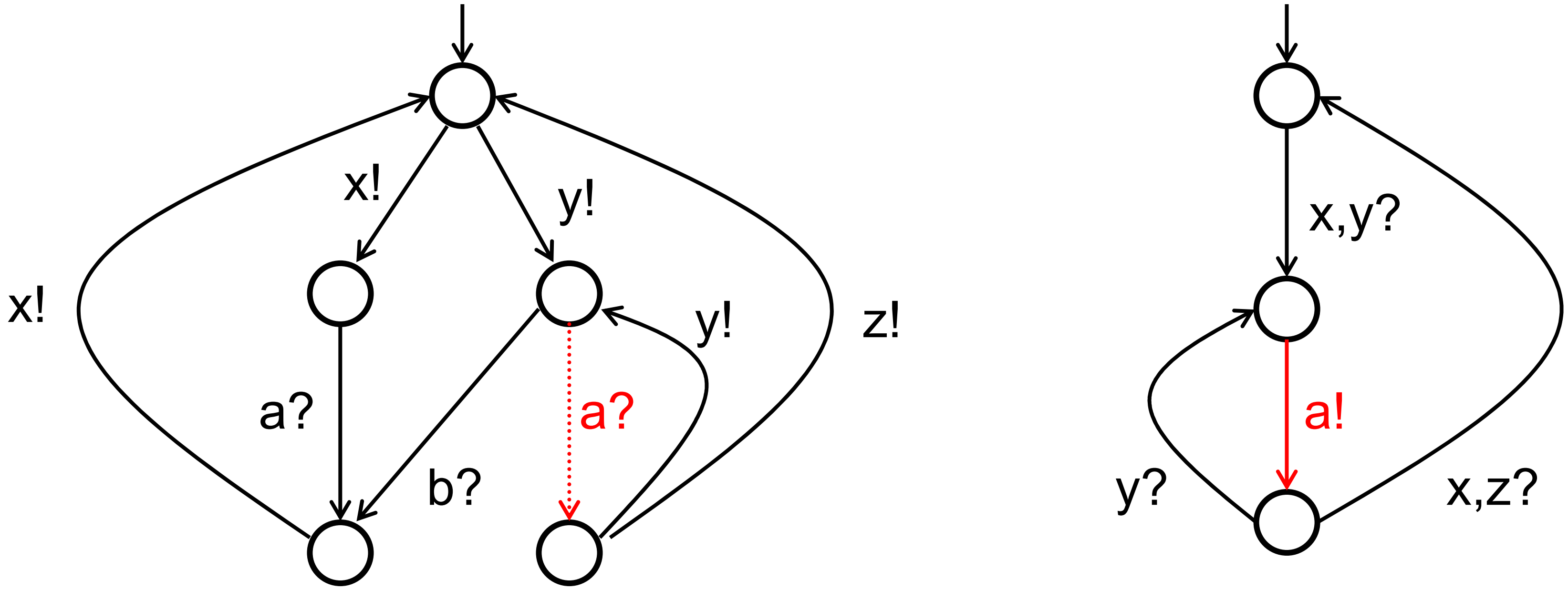
Every legal environment should be a legal environment of the refined process.

Refinement



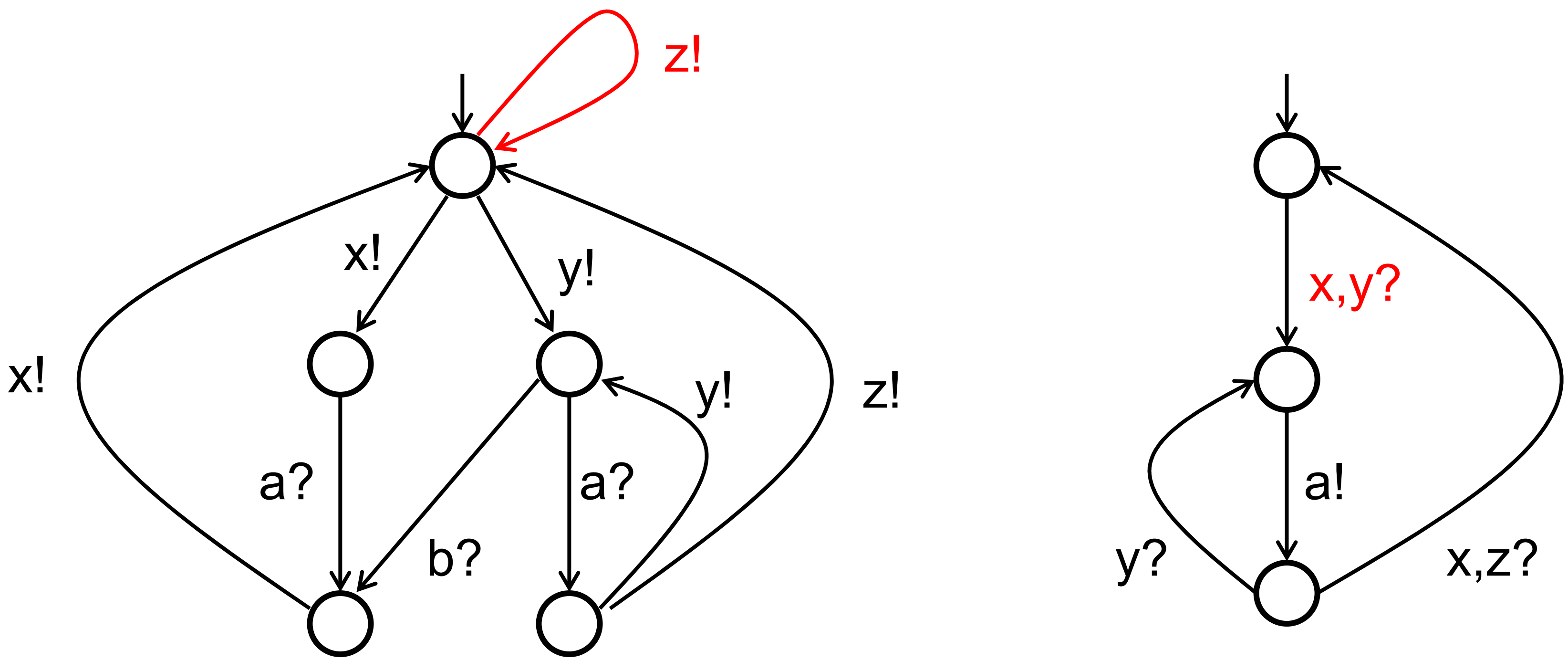
Every legal environment should be a legal environment of the refined process.

Refinement



Every legal environment should be a legal environment of the refined process.

Refinement



Every legal environment should be a legal environment of the refined process.

Interface Refinement: I/O Alternating Simulation

$A' \leq A$

iff

for all outputs o, if A' –o!-> B' , then there exists B such that A –o!-> B and $B' \leq B$.

Interface Refinement: I/O Alternating Simulation

$A' \leq A$

iff

1. for all inputs i, if A –i?-> B , then there exists B′ such that A′ –i?-> B′ and $B' \leq B$,

and

2. for all outputs o, if A′ –o!-> B′ , then there exists B such that A –o!-> B and $B' \leq B$.

Interface Refinement: I/O Alternating Simulation

$A' \leq A$

iff

1. for all inputs i, if A –i?-> B , then there exists B' such that A' –i?-> B' and $B' \leq B$,

and

2. for all outputs o, if A' –o!-> B' , then there exists B such that A –o!-> B and $B' \leq B$.

Every environment (*i.e.,* input strategy that avoids deadlock) for A is an environment for A' [Alur/H/Kupferman/Vardi].

The Principle of Independent Implementability

If A and B is are compatible and $A' \leq A$ and $B' \leq B$, then *A' and B' are compatible* and $A'||B' \leq A||B$.

$A' \leq A$ … A' refines / implements A

The Principle of Independent Implementability

If A and B is are compatible and $A' \leq A$ and $B' \leq B$, then *A' and B' are compatible* and $A'||B' \leq A||B$.

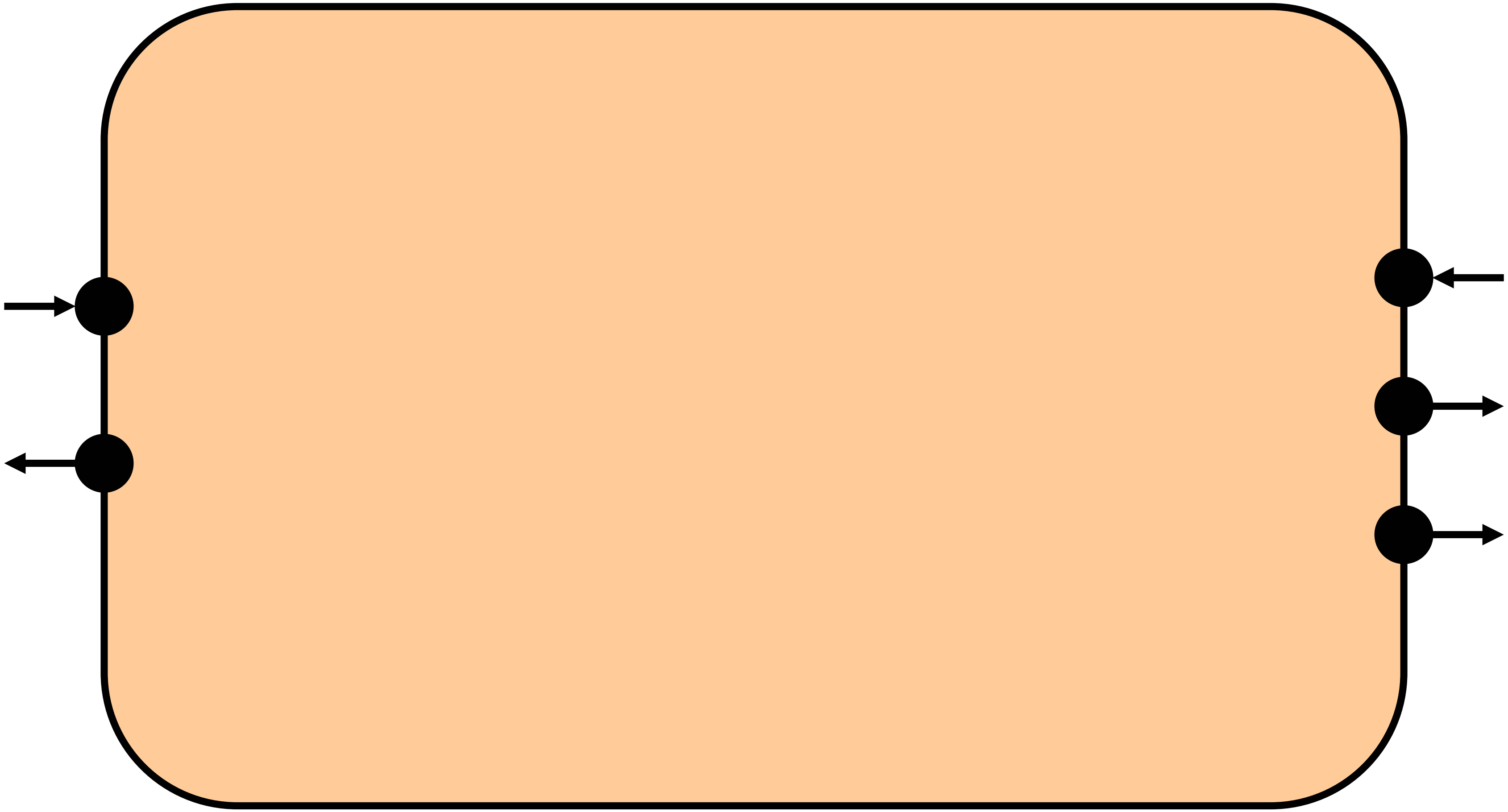
$A' \leq A$ … A' refines / implements A

This is a theorem if

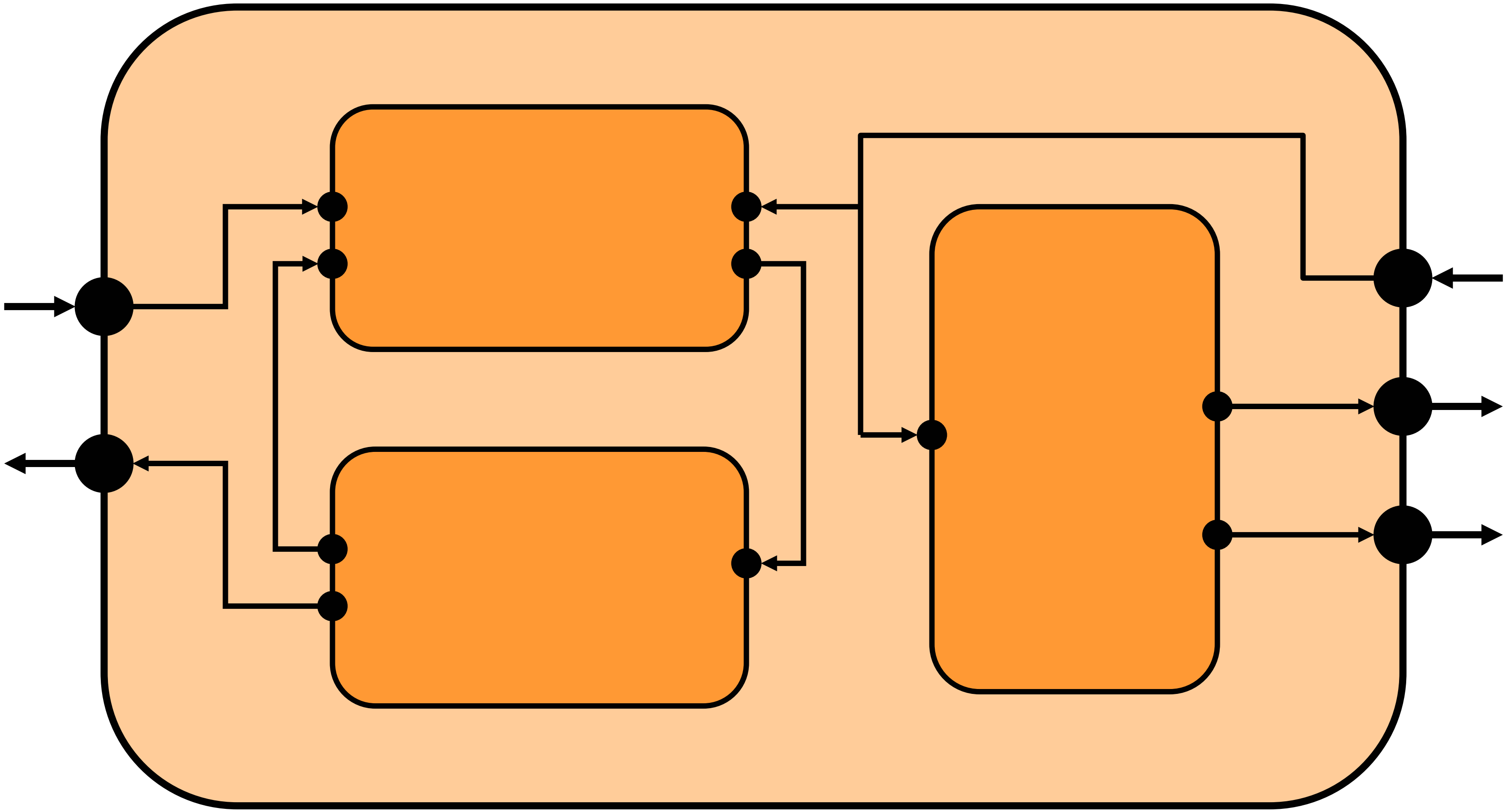
-A, B, A', B' are two-player games Input vs. Output

-two games are compatible if player Input has a winning strategy in the composite game

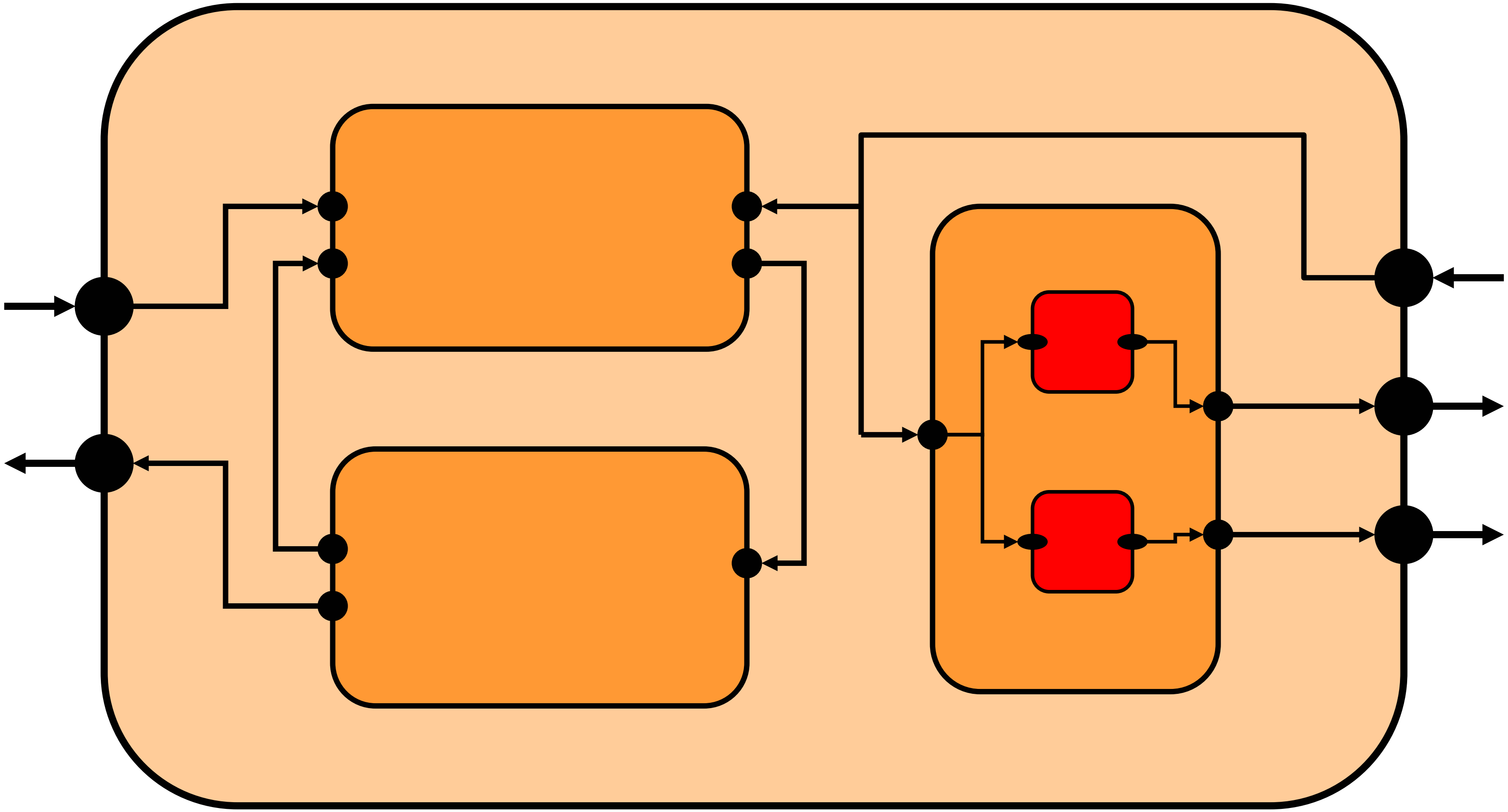
Interface-based Design



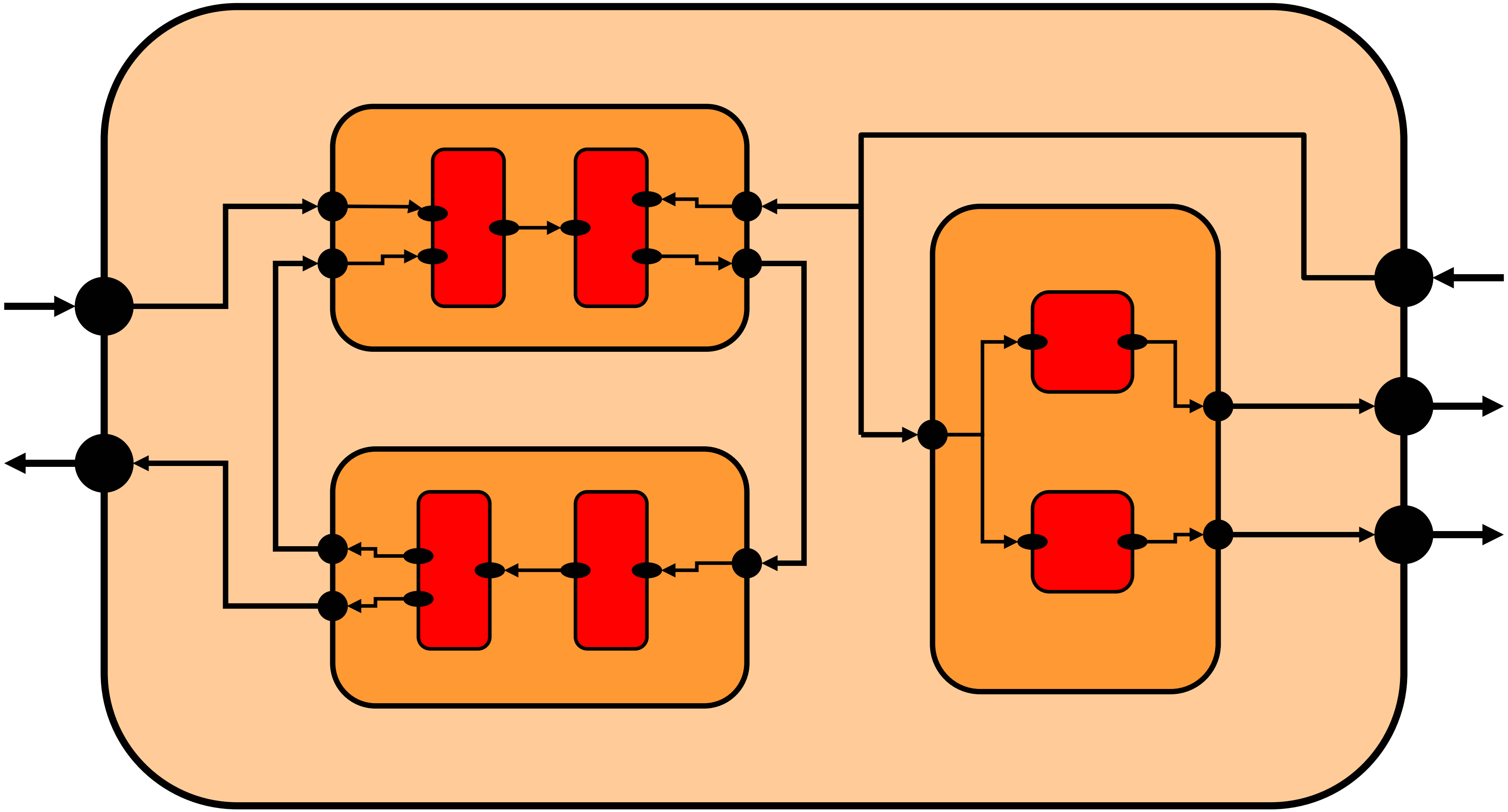
Interface-based Design



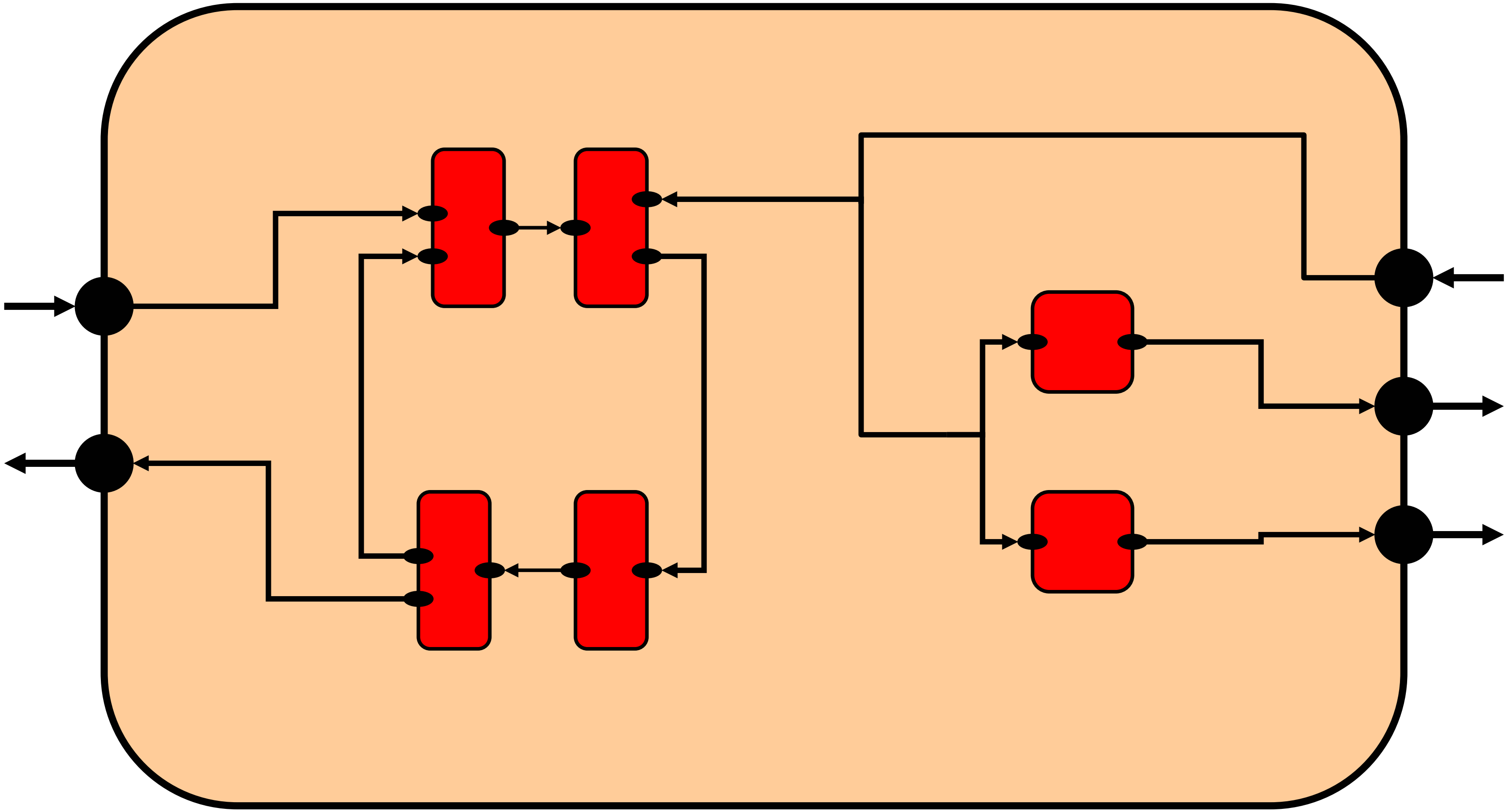
Interface-based Design



Interface-based Design



Interface-based Design



Summary

There are many models of computation (e.g. pushdown, timed, stochastic) and many models of interaction (e.g. synchronous).

Similarly, there are many variants of games (e.g. concurrent vs. turn-based moves; pure vs. randomized strategies).

Summary

There are many models of computation (e.g. pushdown, timed, stochastic) and many models of interaction (e.g. synchronous).

Similarly, there are many variants of games (e.g. concurrent vs. turn-based moves; pure vs. randomized strategies).

The technical details are different, but to ask and answer the kind of questions we discussed, the only important feature of a model is the presence of multiple players.

References

Interface Automata: de Alfaro, H FSE 2001
Secure Equilibria: Chatterjee, H, Jurdzinski LICS 2004