Challenges and Promises in Reactive Modeling:
A Personal Perspective

Tom Henzinger

IST AUSTRIA
Institute of Science and Technology

erc European Research Council

What happens if you push that button?

Formal Methods for Complex Reactive Systems

1. Model design
2. Model analysis

Some Paradigmatic Modeling Languages

Moore/Mealy machines
Petri nets
Kahn networks
Threads
CSP/CCS
Synchronous languages
Statecharts
Timed automata
Pi-calculus
etc.

Some Personal Modeling Languages

Hybrid automata: mixing discrete and continuous behavior
Reactive modules: mixing synchronous and asynchronous interaction
Interface automata: independent implementability of components
Giotto: time-triggered task coordination

Some Personal Modeling Languages

Hybrid automata: mixing discrete and continuous behavior
Reactive modules: mixing synchronous and asynchronous interaction
Interface automata: independent implementability of components
Giotto: time-triggered task coordination

Some Personal Specification Languages

Game temporal logics: specifying cooperation and competition
Nested weighted automata: specifying resource consumption
Pre/post fixpoint calculus: specifying symbolic state-space exploration

More and Less Solved Issues in Language Design

Concurrency
Mobility

Modularity
Strategic aspects
Quantitative aspects

More and Less Solved
Modularity Aspects in Language Design

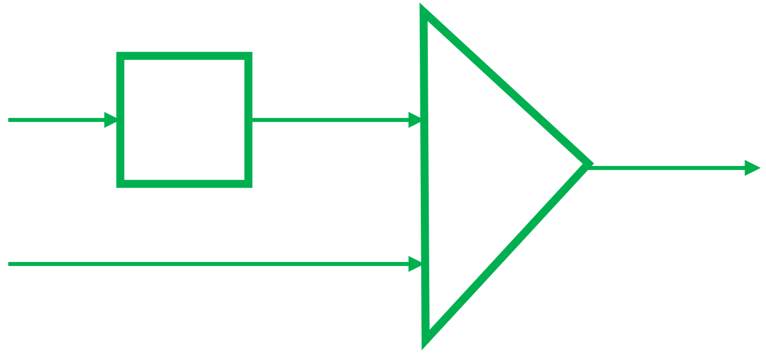
Composition and communication
Spatial abstraction

Temporal abstraction
Shared refinement
Mixing data flow and control flow

Graphical System Models

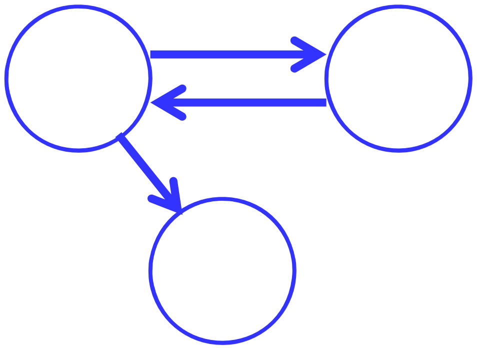
Engineering

Component model: transfer equation
Composition: spatial (parallel)
Connection: data flow



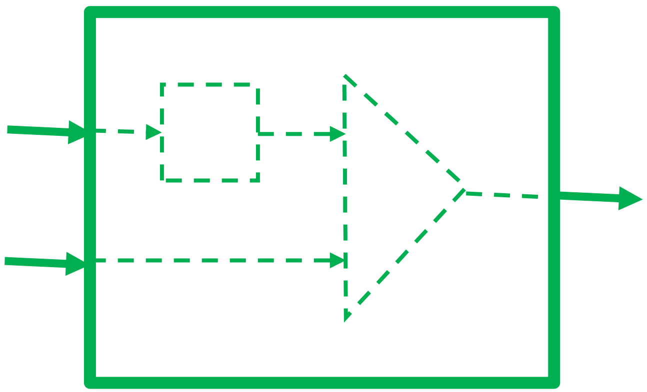
Computer Science

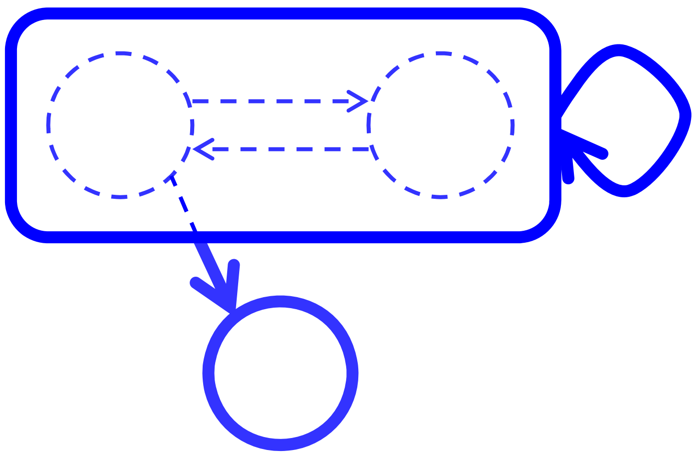
Component model: subroutine
Composition: temporal (sequential)
Connection: control flow



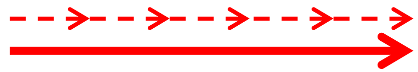
Abstraction Hierarchies

Spatial

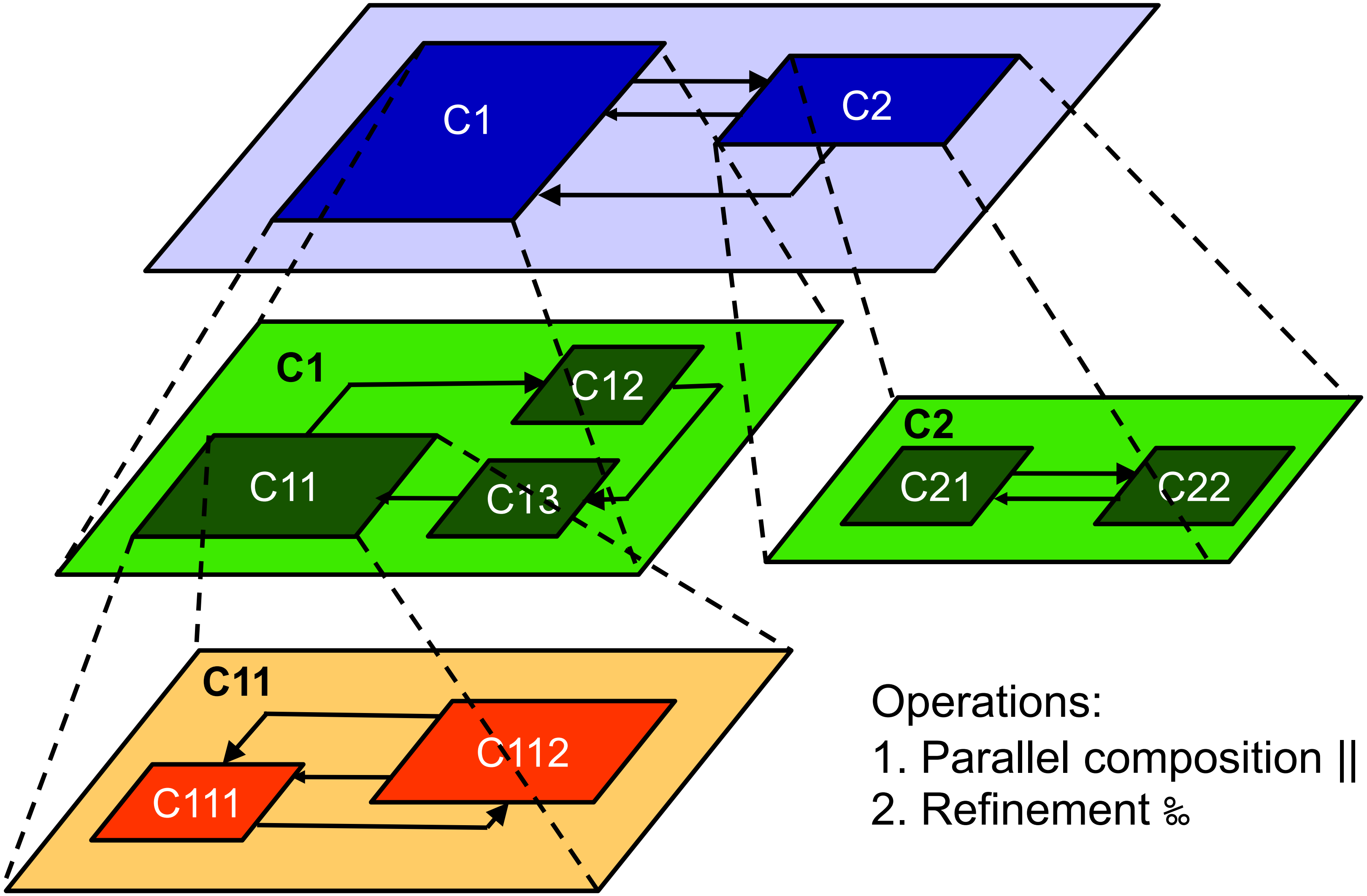




Temporal

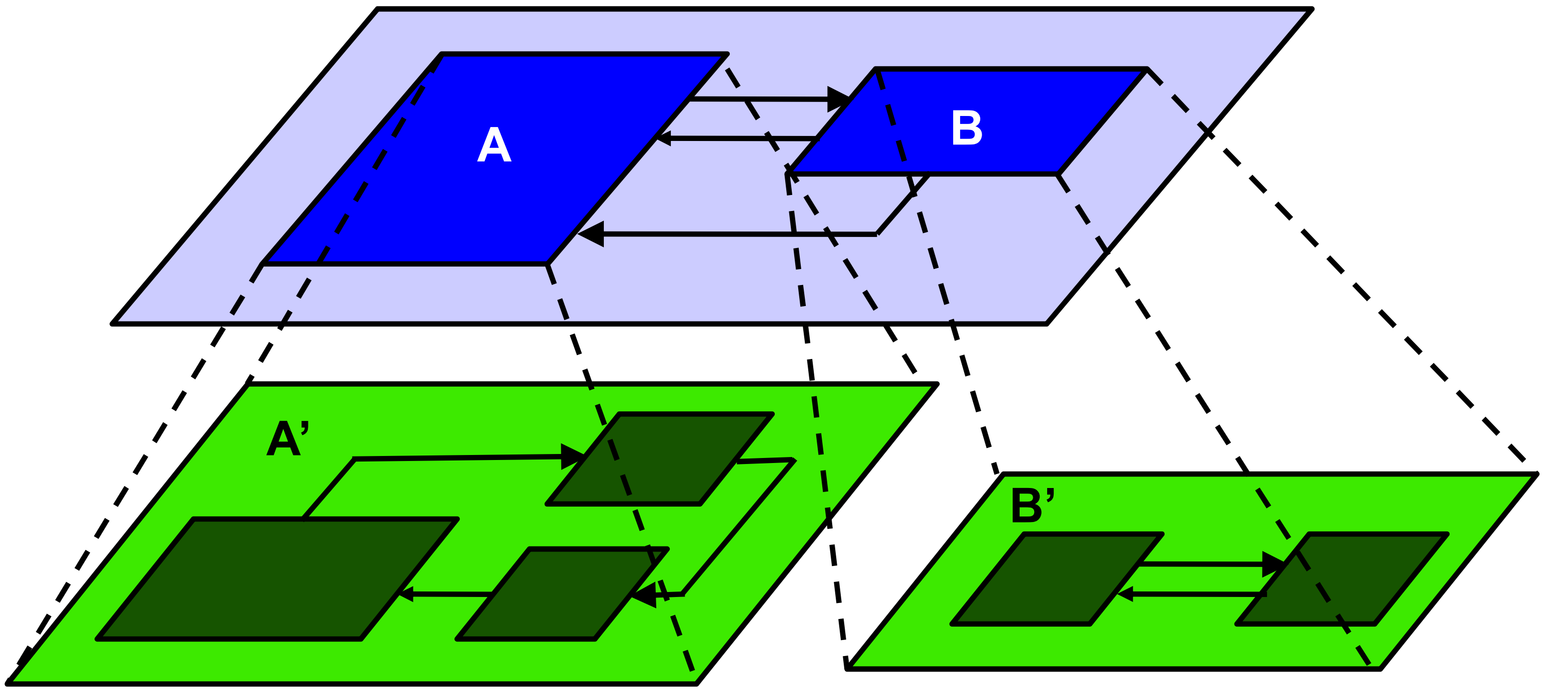


Top-down Design

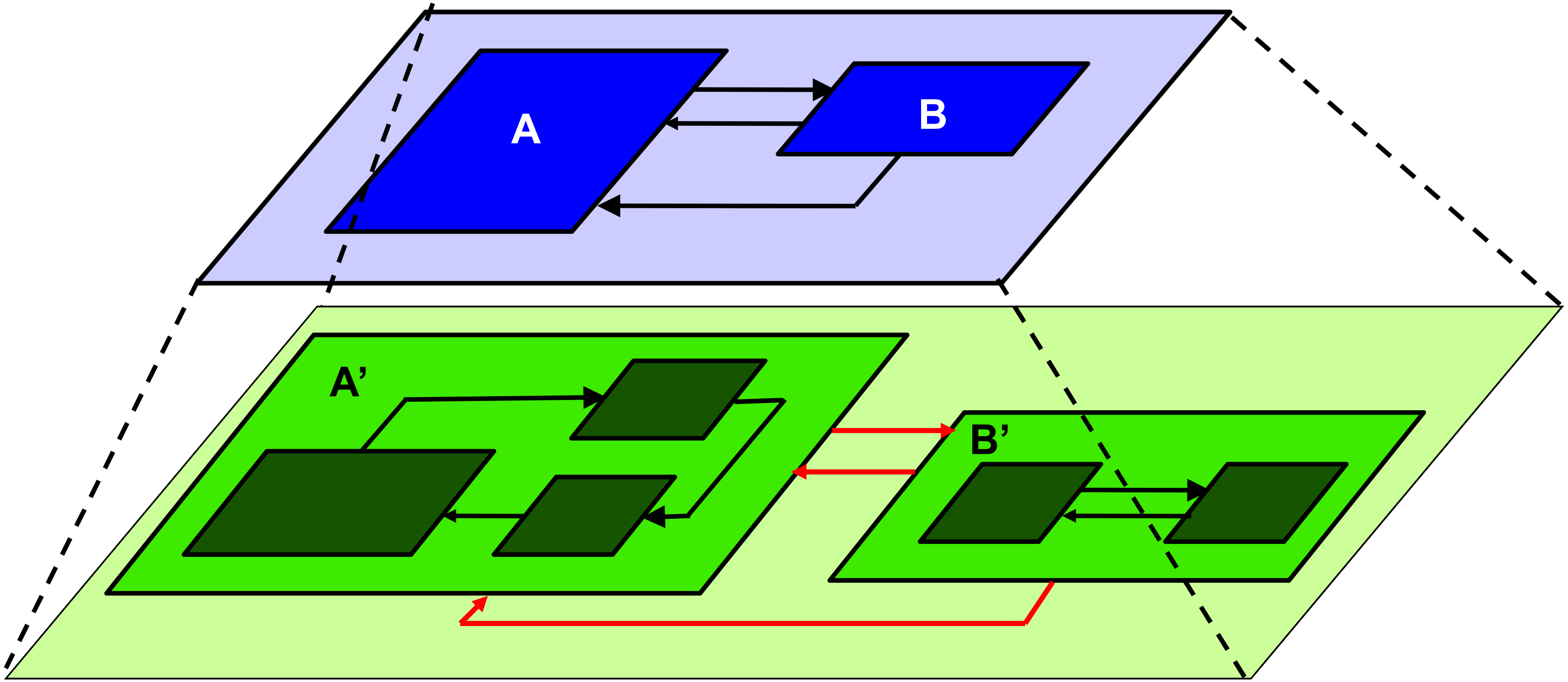


Operations:
1. Parallel composition ||
2. Refinement ‰

Independent Implementability

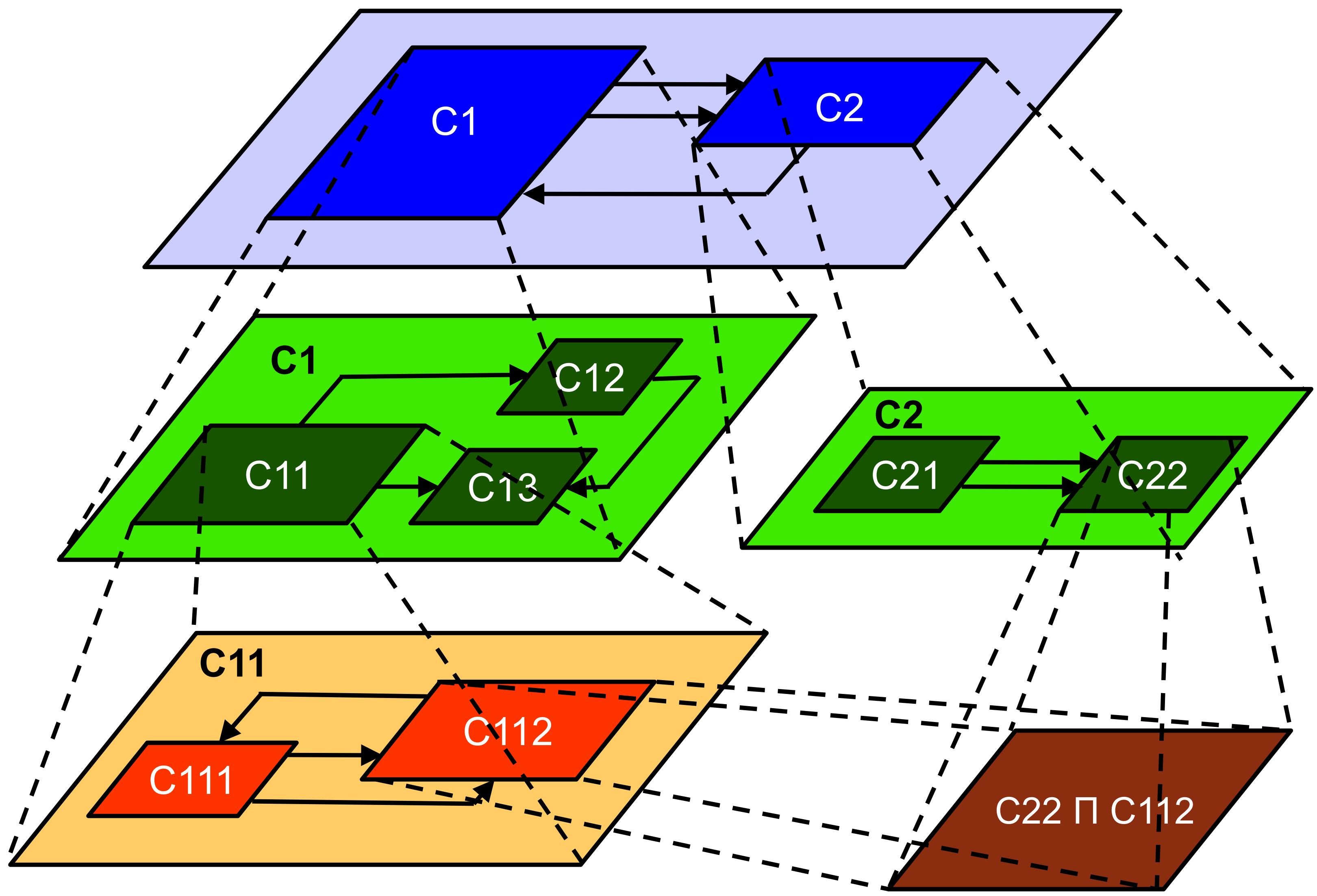


Independent Implementability

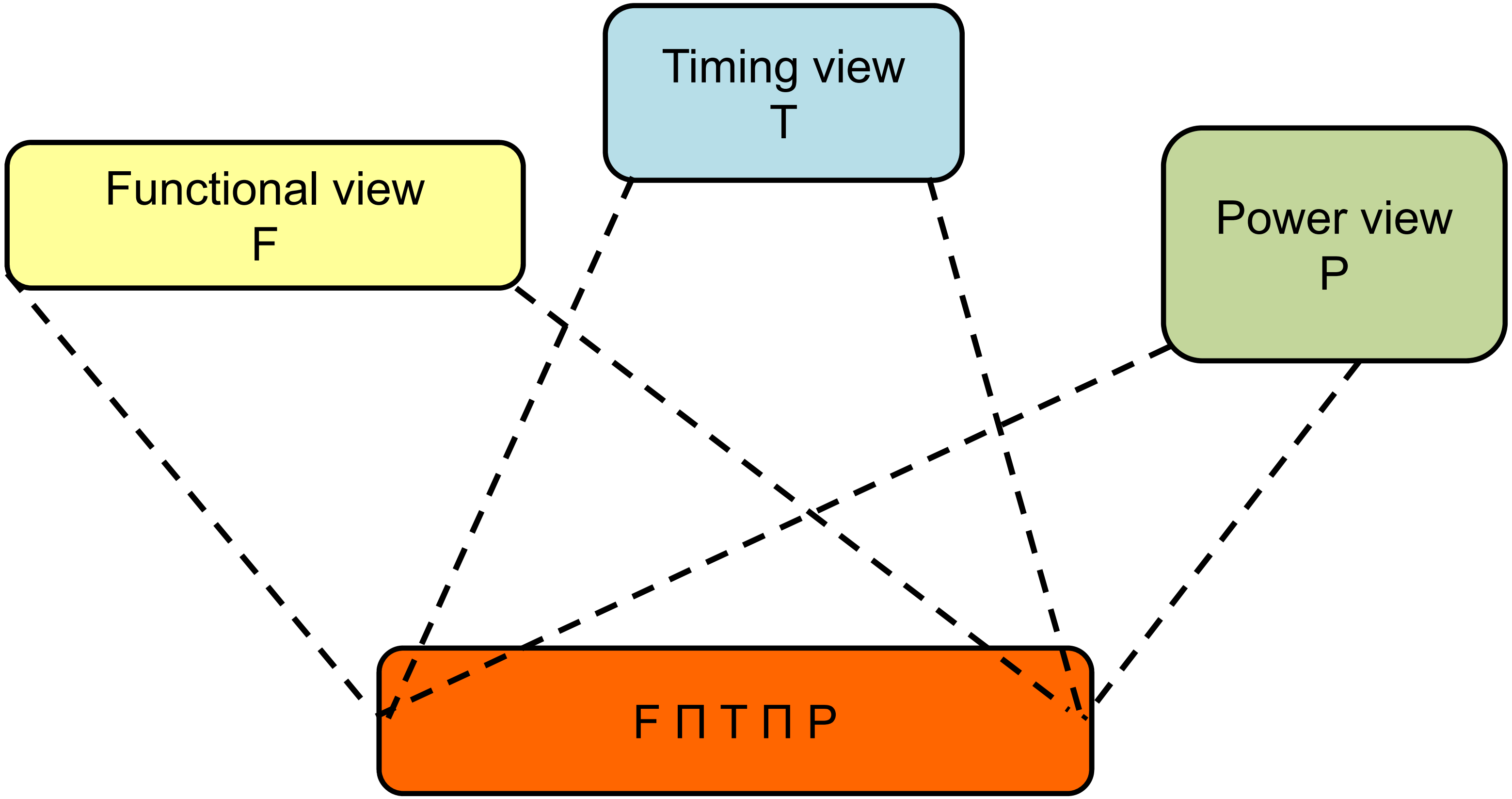


If A and B are composable and A' ≤ A and B' ≤ B,
then A' and B' are composable and A'||B' ≤ A||B.

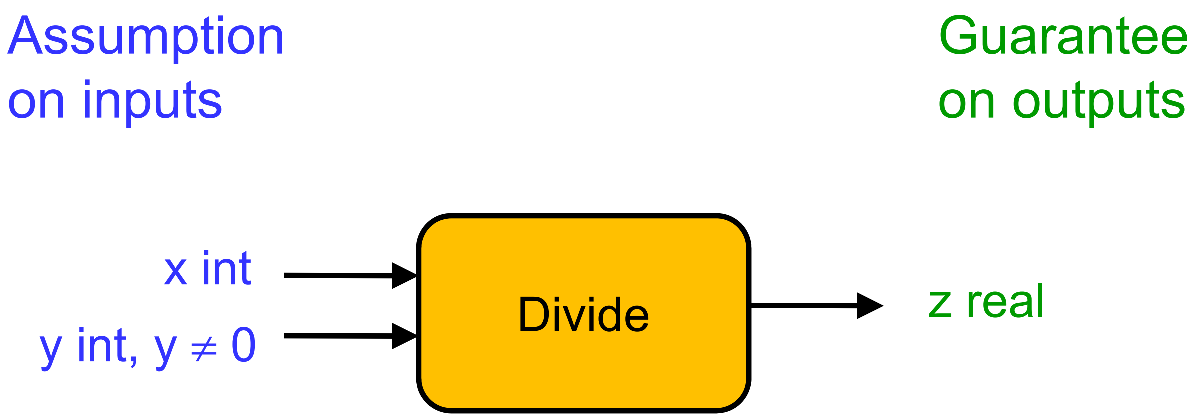
Component-based Design: Shared Refinement



Multiple Viewpoints: Shared Refinement

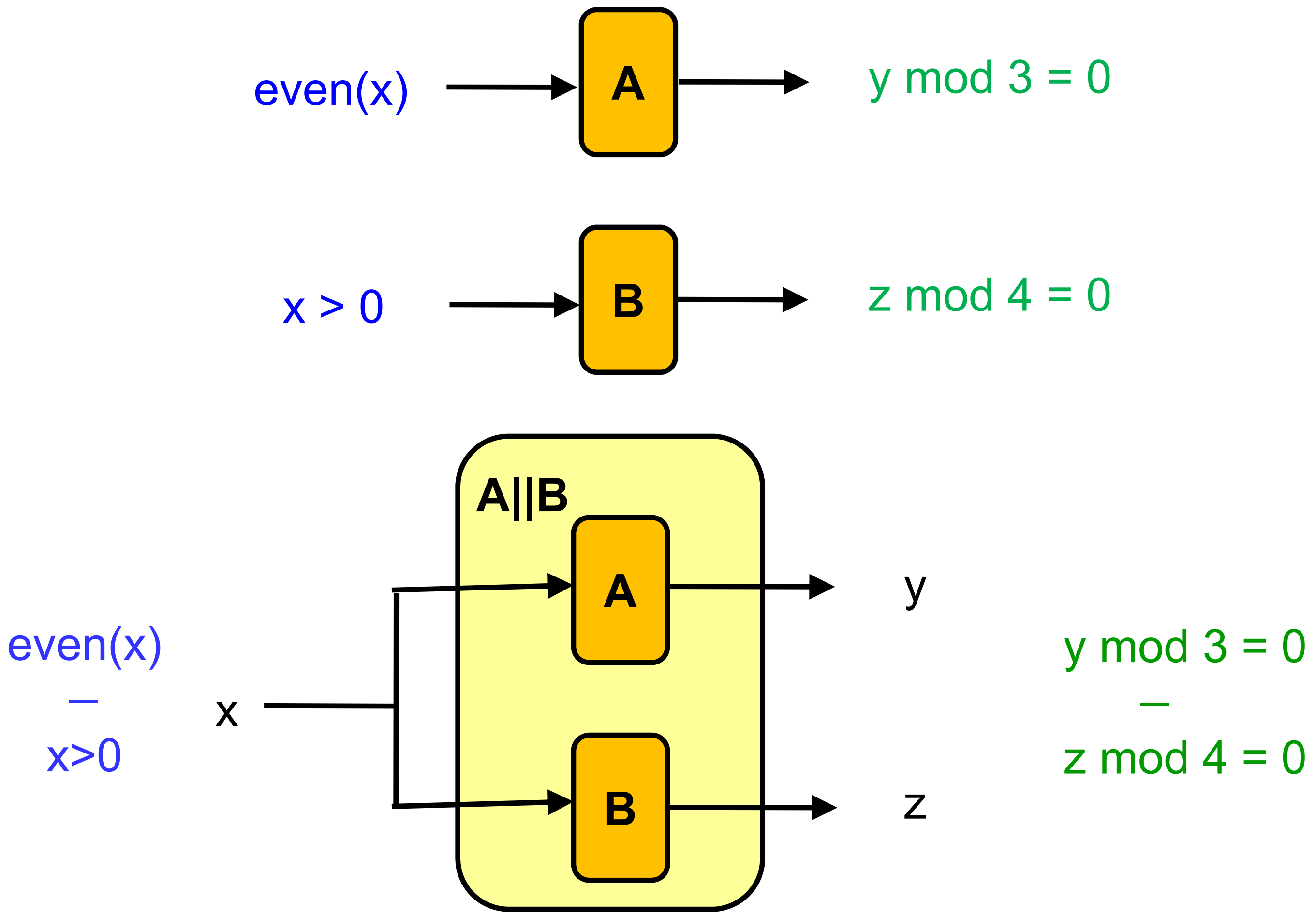


Stateless Interface

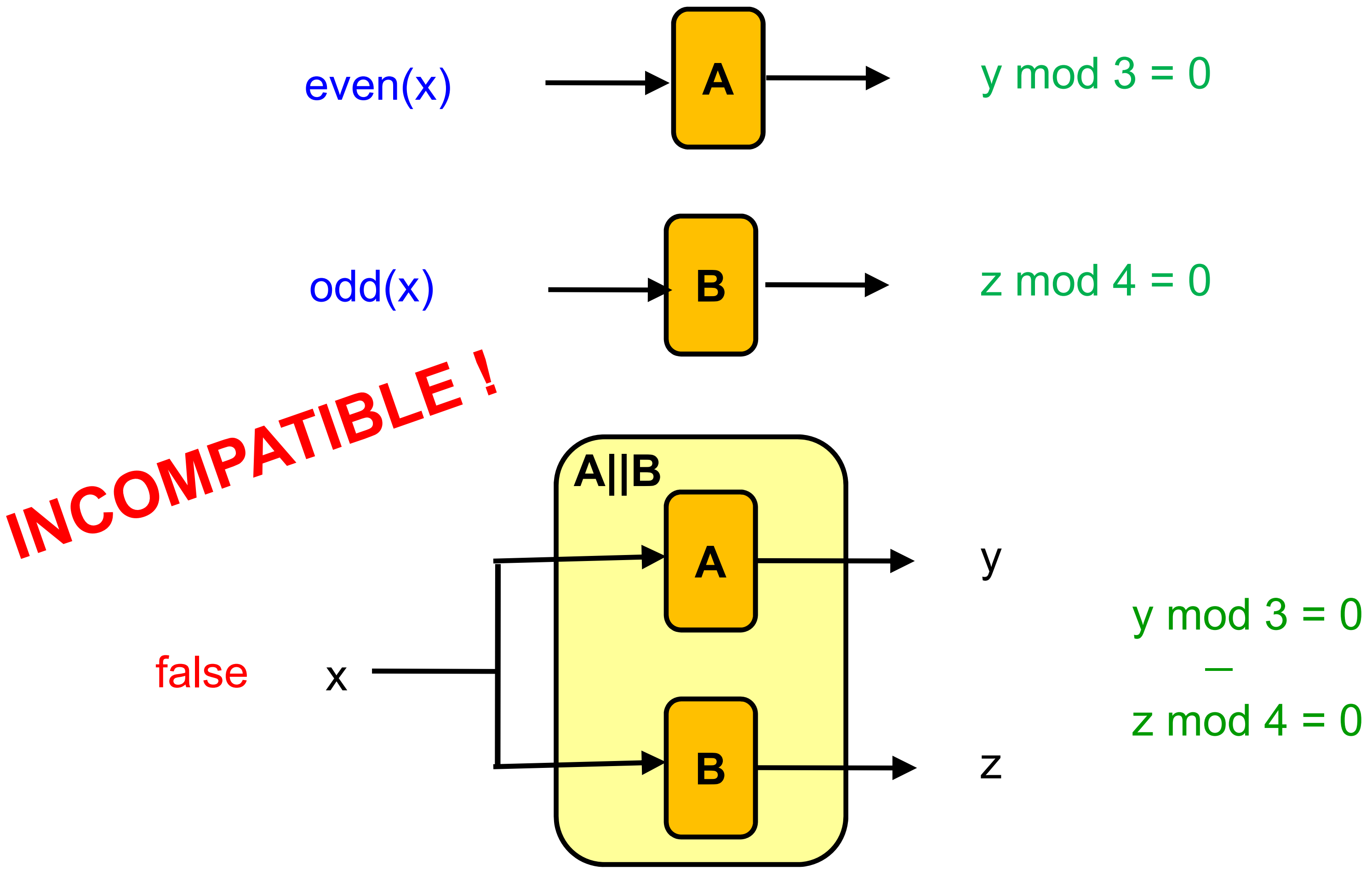


Well-formedness:
1. Assumption satisfiable
2. Guarantee satisfiable

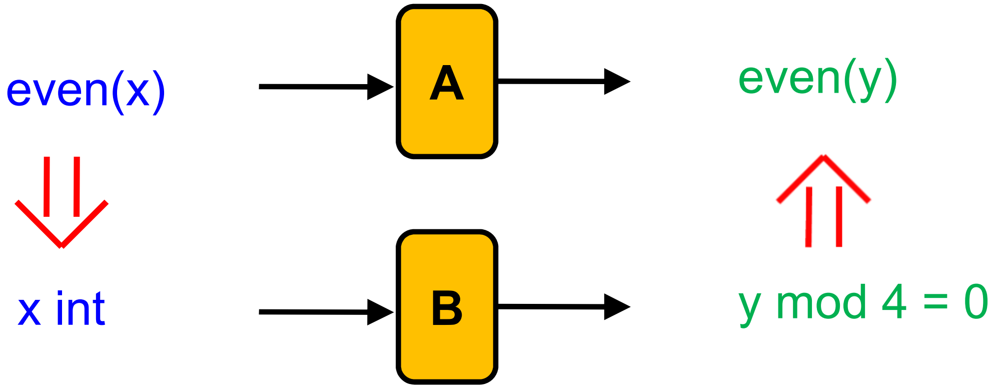
Parallel Composition



Parallel Composition

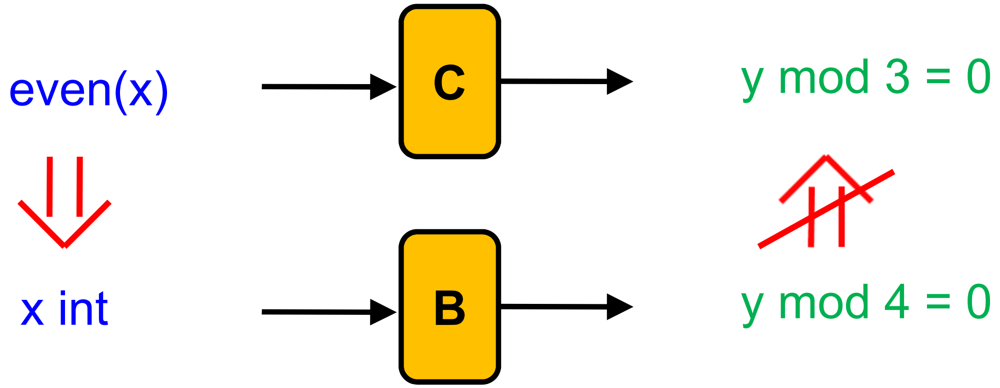


Refinement



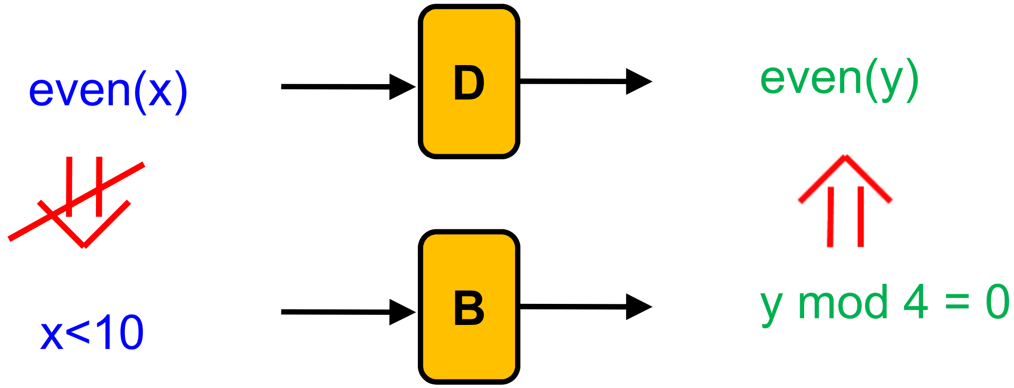
B refines A

Refinement



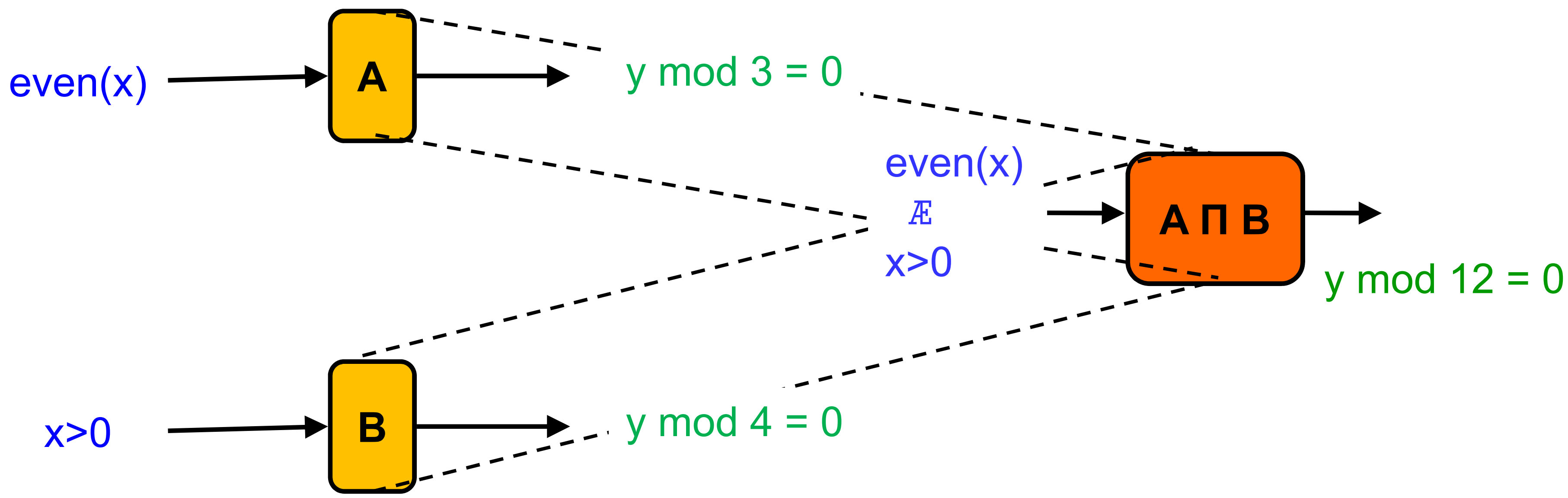
B does not refine C:
Implementation may produce inadmissible outputs.

Refinement



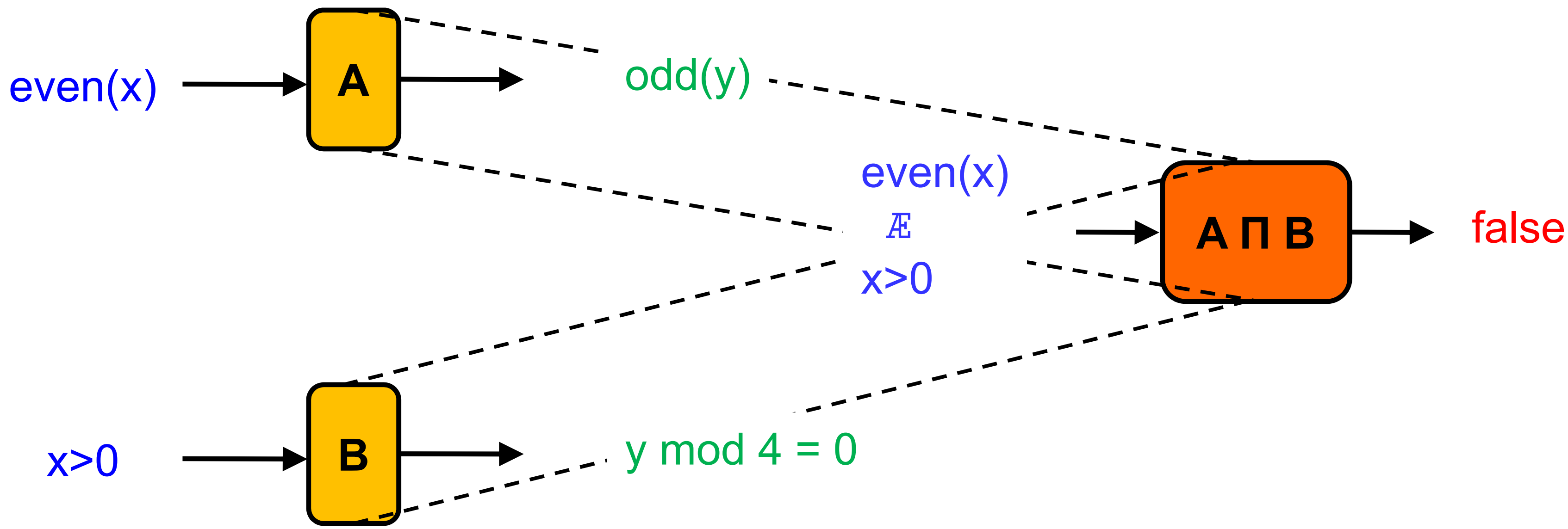
B does not refine D:
Implementation does not accept all admissible inputs.

Shared Refinement



A Π B can be used in any design as an implementation of A, and as an implementation of B.

Shared Refinement



NOT SHARED-REFINABLE !

Modularity Challenge:

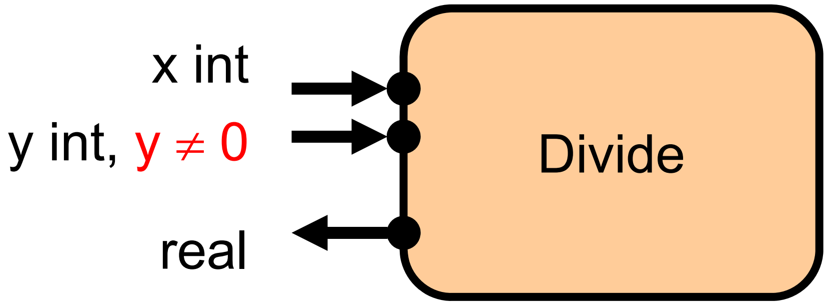
Define a modeling language with feedback,
independent implementability, and shared refinement.

More and Less Solved
Strategic Aspects in Language Design

Controller synthesis

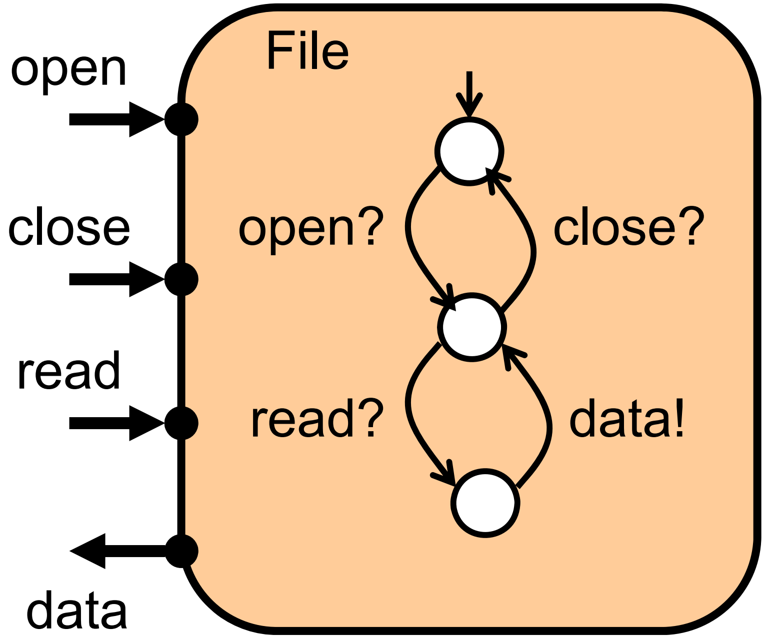
Competitive/cooperative composition
Strategy-preserving abstraction
Behavioral equilibria

Stateless Interface



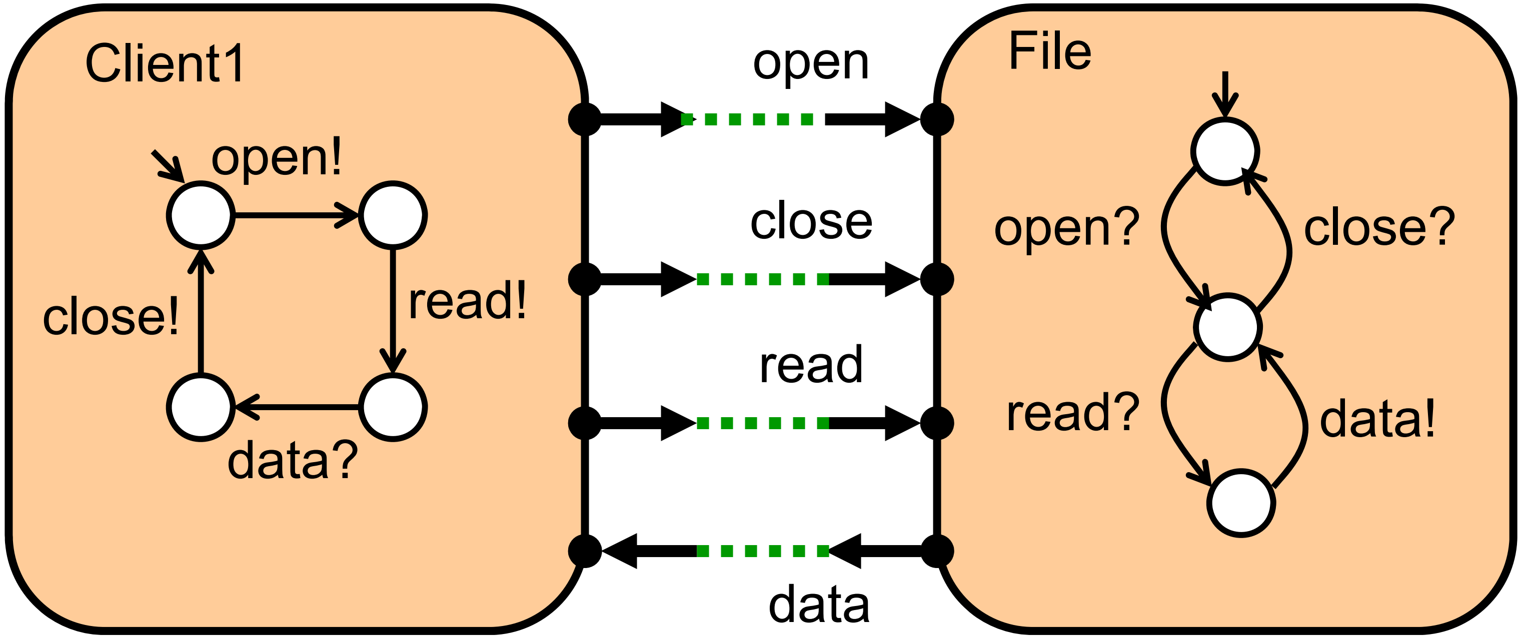
This interface constrains the client's data.

Stateful Interface

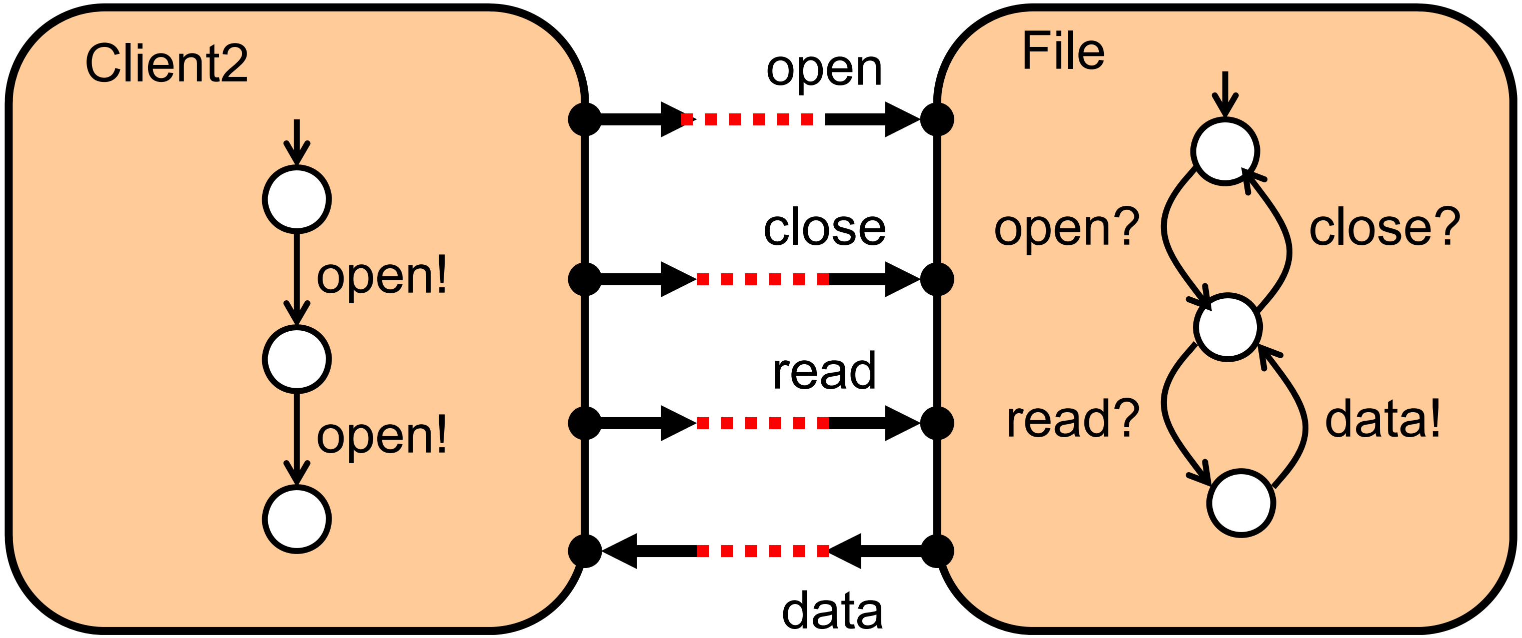


This interface constrains the client's control.

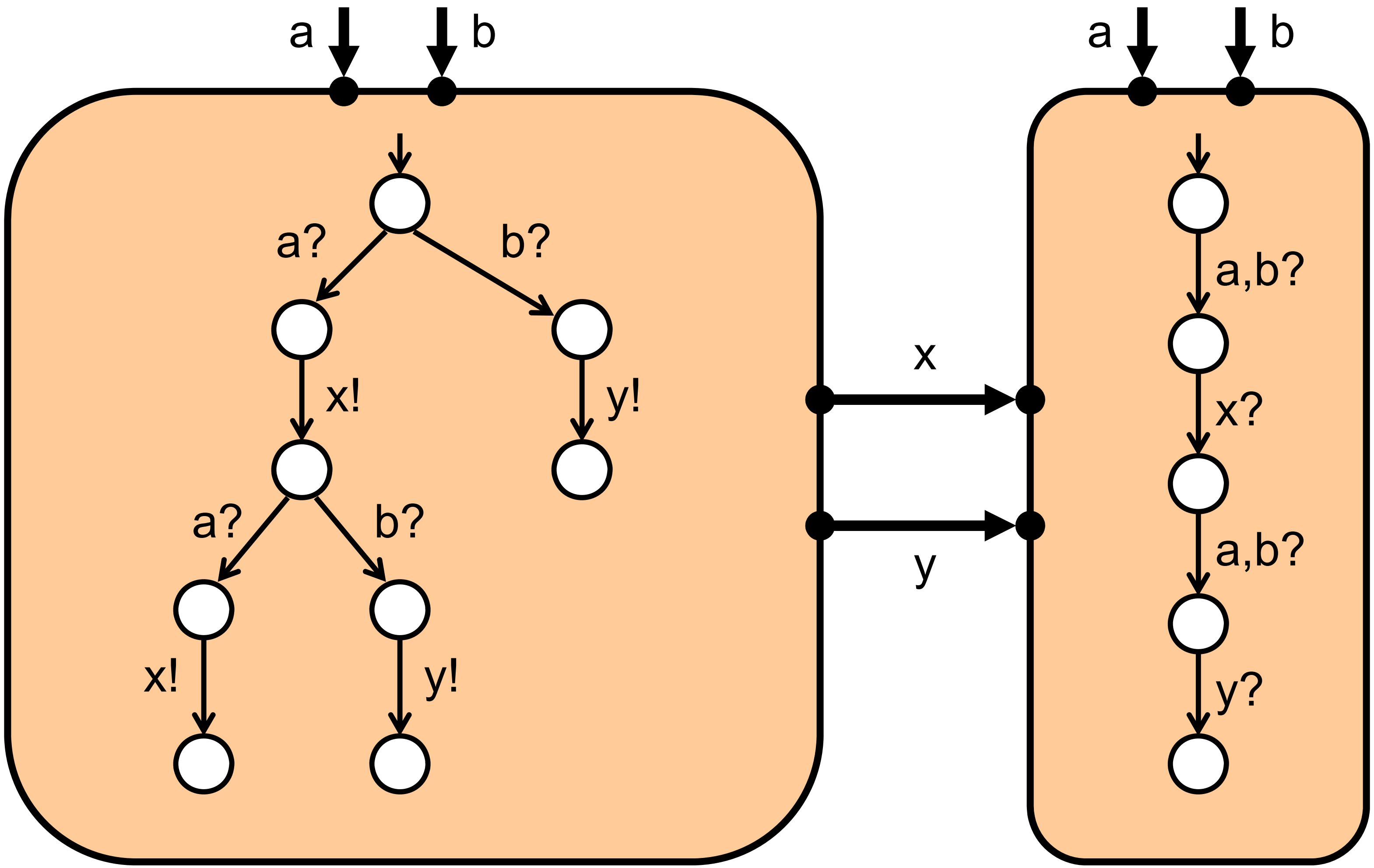
Stateful Interface Compatibility



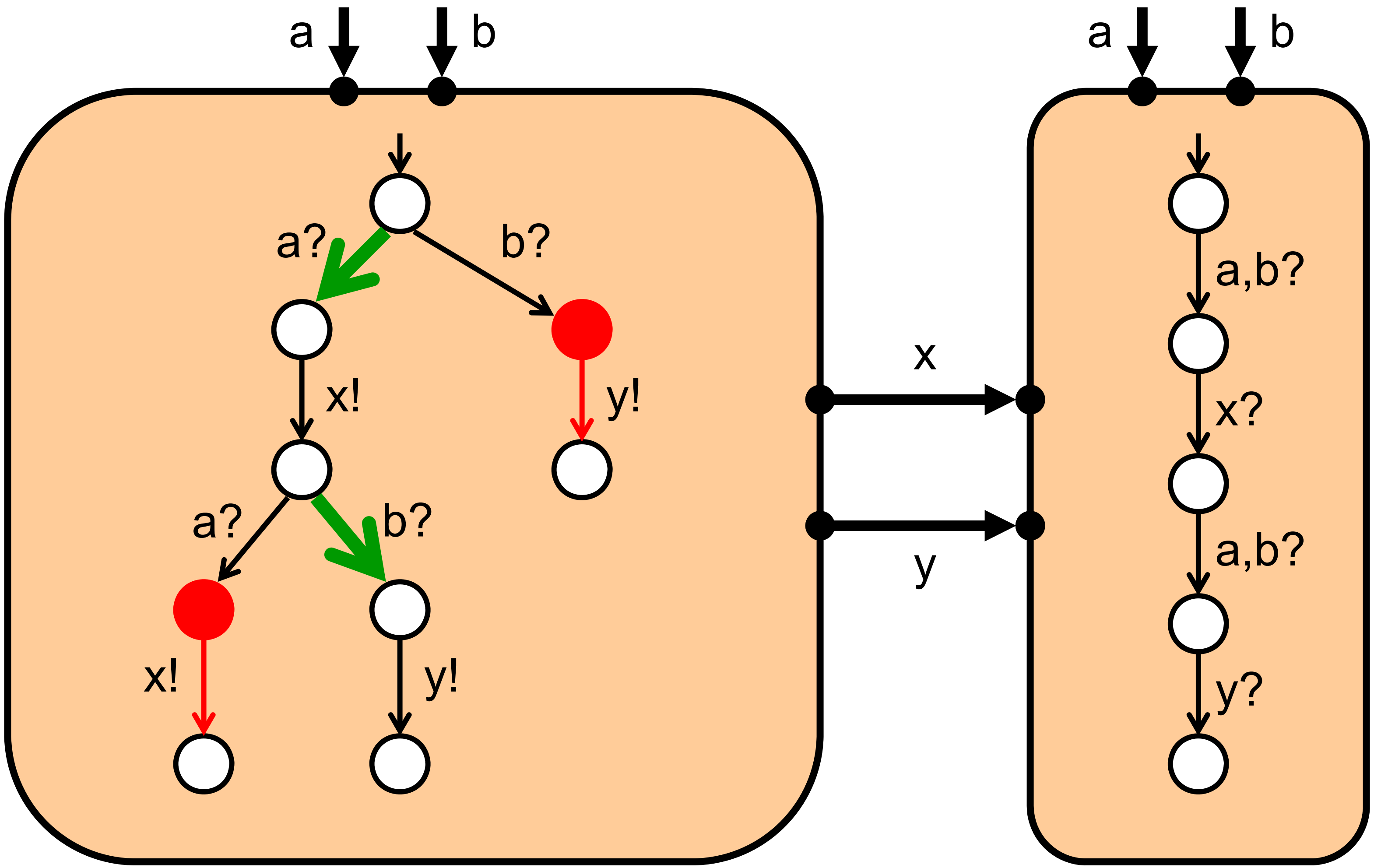
Stateful Interface Incompatibility



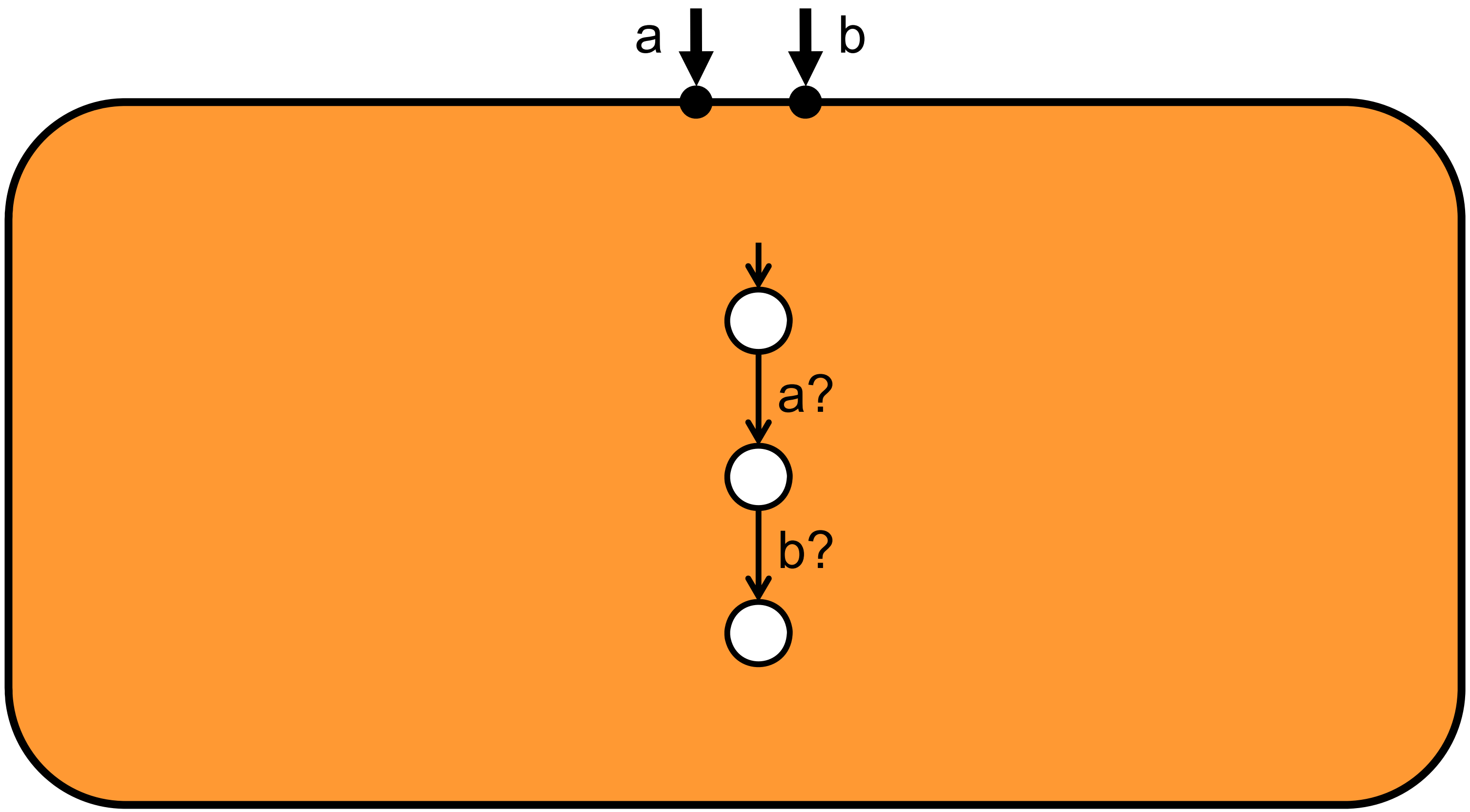
Stateful Interface Composition



Stateful Interface Composition



Composition as Game: Composite Interface



Two Threads

P_1:

init x := 0

loop

    choice
        | x := x+1 mod 2
        | x := 0

    end choice

end loop

P_2:

init y := 0

loop

    choice
        | y := x
        | y := x+1 mod 2

    end choice

end loop

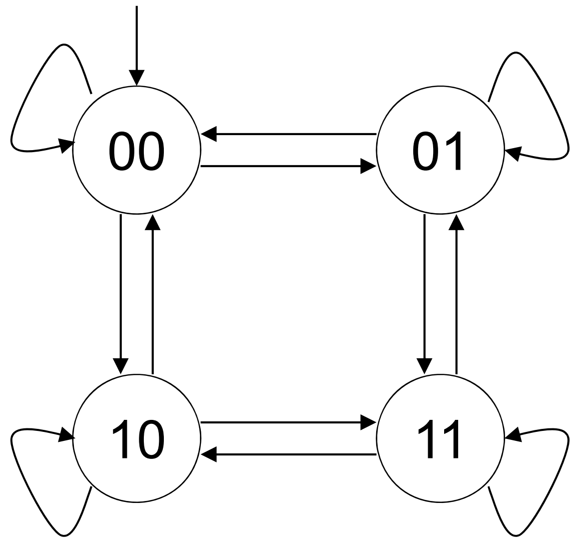
Φ_1: □ (x ¿ y)

Φ_2: □ even(y)

Graph Questions

X ;#□ (x ¿ y)

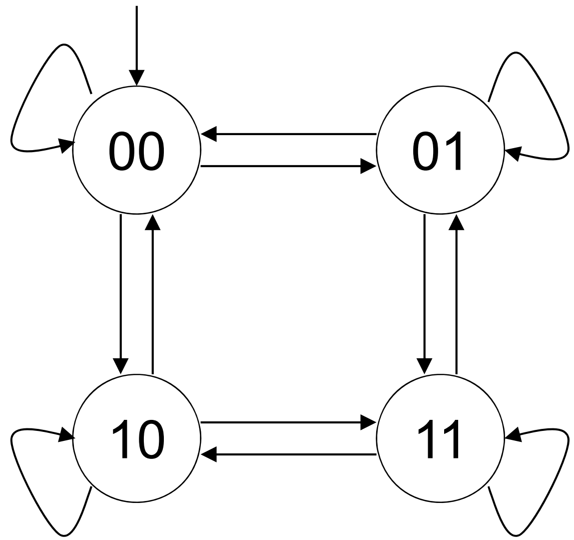
✓ <#□ (x ¿ y)



Game Questions

X ;#□ (x ¿ y)

✓ <#□ (x ¿ y)



kkP$_1$ll □ (x ¿ y)

kkP$_2$ll □ even(y)

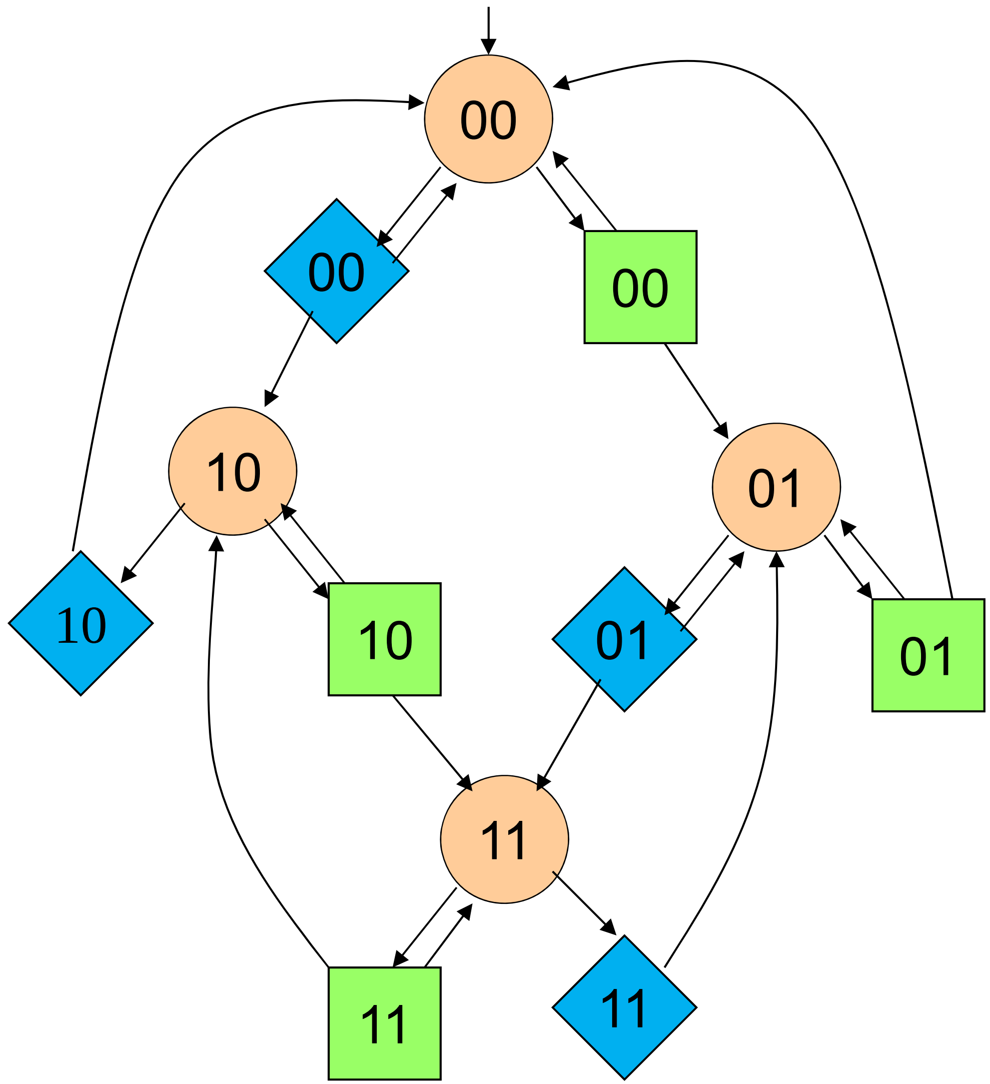
Game Questions

X ;#□ (x ¿ y)

✓ <#□ (x ¿ y)

X kkP$_1$ll □ (x ¿ y)

✓ kkP$_2$ll □ even(y)

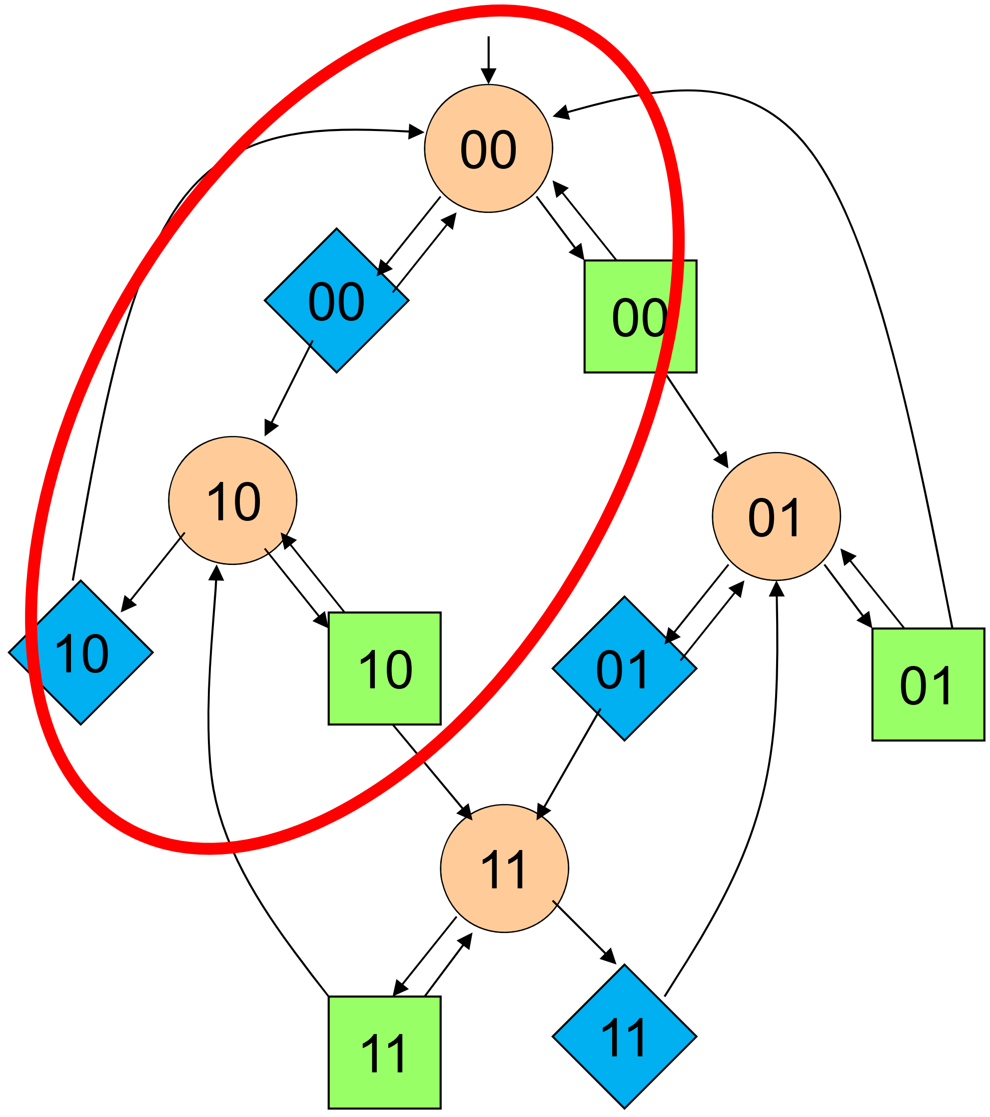


Equilibrium Question

kkP_1ll □ (x ¿ y)

✓ ⊗

kkP_2ll □ even(y)



Synthesis Problems

Controller Synthesis

Given: process P, specification Φ, and environment E

Find: refinement P' of P so that P'||E satisfies Φ

Solution: P' = winning strategy in game P against E for objective Φ

Synthesis Problems

Controller Synthesis

Given: process P, specification Φ, and environment E

Find: refinement P' of P so that P'||E satisfies Φ

Solution: P' = winning strategy in game P against E for objective Φ

Co-synthesis

Given:

-two processes P_1 and P_2

-specifications Φ_1 and Φ_2 for P_1 and P_2

Find:

refinements P'_1 and P'_2 of P_1 and P_2 so that

$P'_1||P'_2||S$ satisfies $\Phi_1 \wedge \Phi_2$ for every fair scheduler S

Mutual Exclusion

```
while( true ) {
  flag[1] := true; turn := 2;

  choice
  | while( flag[1] ) nop;
  | while( flag[2] ) nop;
  | while( turn=1 ) nop;
  | while( turn=2 ) nop;
  | while( flag[1] & turn=2 ) nop;
  | while( flag[1] & turn=1 ) nop;
  | while( flag[2] & turn=1 ) nop;
  | while( flag[2] & turn=2 ) nop;
  end choice;

  CS1 := true; fin_wait;
  CS1 := false; arbitrary_wait;
}
```

```
while( true ) {
  flag[2] := true; turn :=1;

  choice
  | while( flag[1] ) nop;
  | while( flag[2] ) nop;
  | while( turn=1 ) nop;
  | while( turn=2 ) nop;
  | while( flag[1] & turn=2 ) nop;
  | while( flag[1] & turn=1 ) nop;
  | while( flag[2] & turn=1 ) nop;
  | while( flag[2] & turn=2 ) nop;
  end choice;

   CS2 := true; fin_wait;
   CS2 := false; arbitrary_wait;
}
```

Co-synthesis Formulation 1

Do there exist refinements P'_1 and P'_2 so that
$[P'_1 \mid\mid P'_2 \mid\mid S] \gg (\Phi_1 - \Phi_2)$ for every fair scheduler S ?

Solution: game $P_1 \mid\mid P_2$ against S for objective $\Phi_1 - \Phi_2$

Too weak (solution has P_1 and P_2 alternate).

Co-synthesis Formulation 2

Do there exist refinements P'_1 and P'_2 so that both $[P'_1 \,||\, P_2 \,||\, S] \gg \Phi_1$ and $[P_1 \,||\, P'_2 \,||\, S] \gg \Phi_2$ for every fair scheduler S ?

Solution: two games P_1 against $P_2||S$ for objective Φ_1, and P_2 against $P_1||S$ for objective Φ_2

Too strong (answer is NO).

Co-synthesis Formulation 3

Do there exist refinements P'_1 and P'_2 so that

1. $[P'_1 \parallel P_2 \parallel S] \gg (\Phi_2 \ , \ \Phi_1)$
2. $[P_1 \parallel P'_2 \parallel S] \gg (\Phi_1 \ , \ \Phi_2)$
3. $[P'_1 \parallel P'_2 \parallel S] \gg (\Phi_1 - \Phi_2)$

for every fair scheduler S ?

Mutual Exclusion

```
while( true ) {
  flag[1] := true; turn := 2;

  while( flag[2] & turn=1 ) nop;

  CS1 := true; fin_wait;
  CS1 := false; arbitrary_wait;
}
```

```
while( true ) {
  flag[2] := true; turn := 1;

  while( flag[1] & turn=2 ) nop;

  CS1 := true; fin_wait;
  CS1 := false; arbitrary_wait;
}
```

Solution is exactly Peterson's mutual-exclusion protocol.

Strategic Synthesis Challenge:

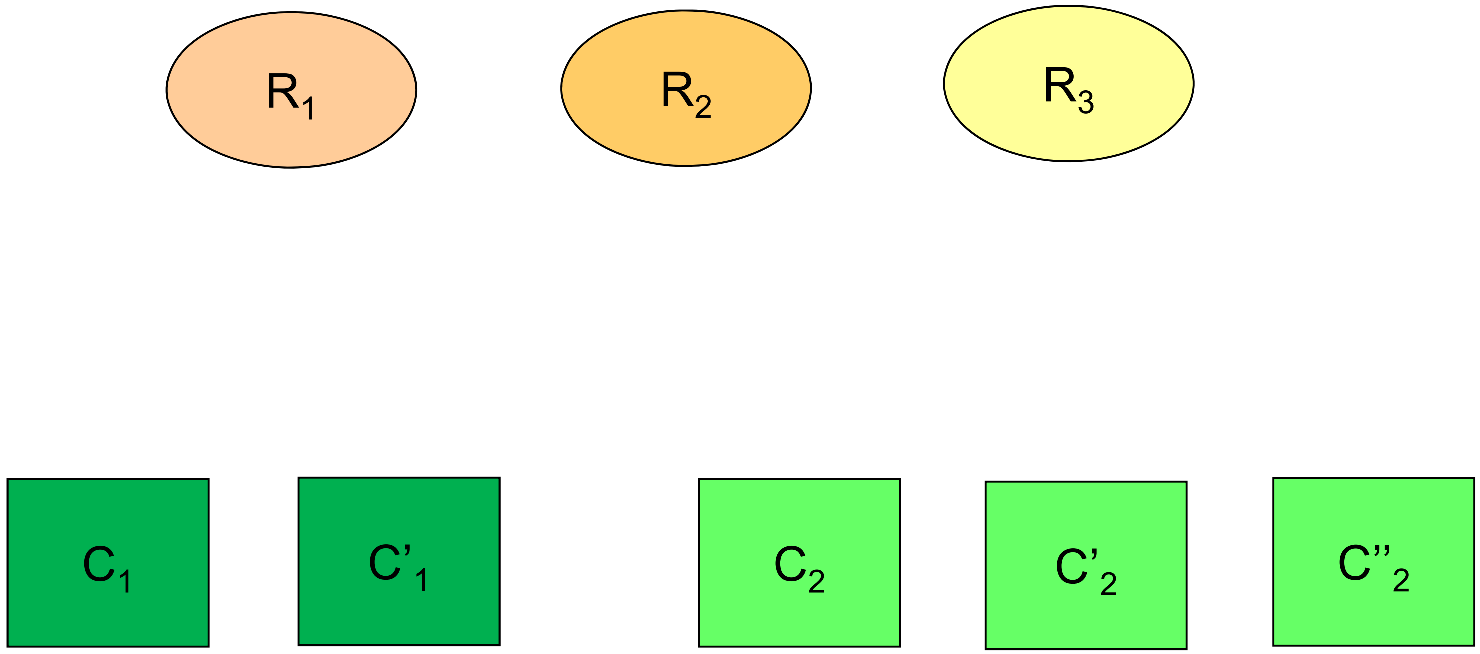
Solve co-synthesis for more than two components.

More and Less Solved
Quantitative Aspects in Language Design

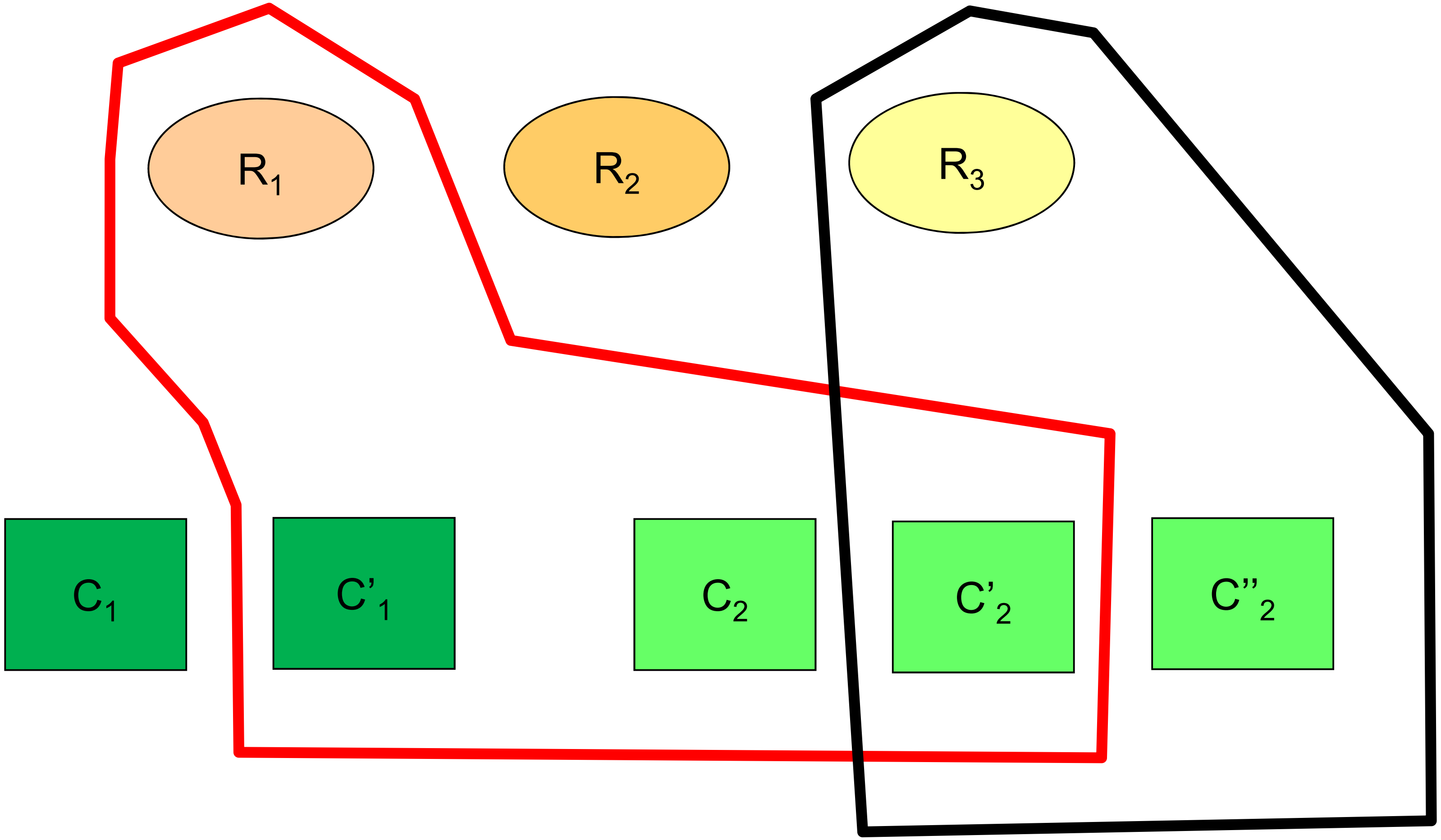
Real time
Probabilistic choice

Reliability and robustness
Average resource consumption
Behavioral metrics

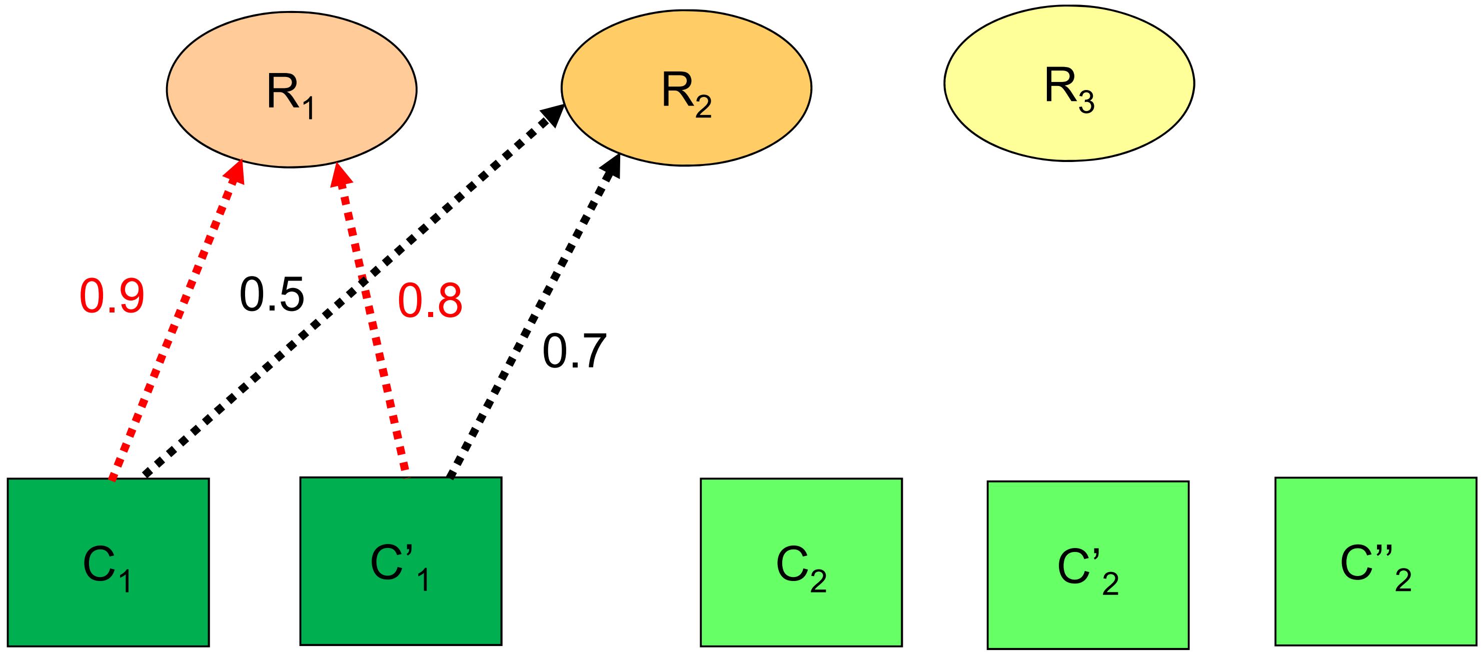
Components and Requirements



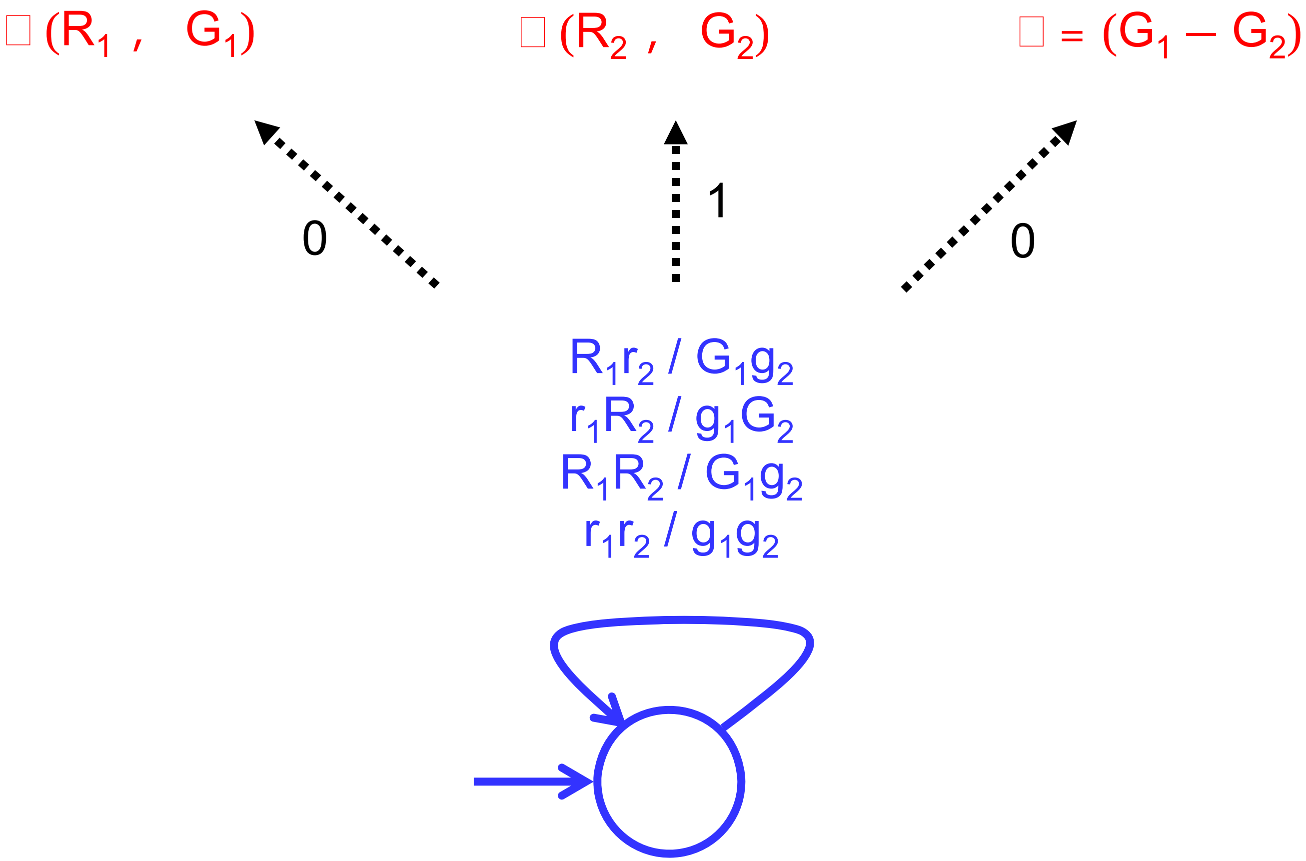
Correctness = Preorder



Fitness = Metric

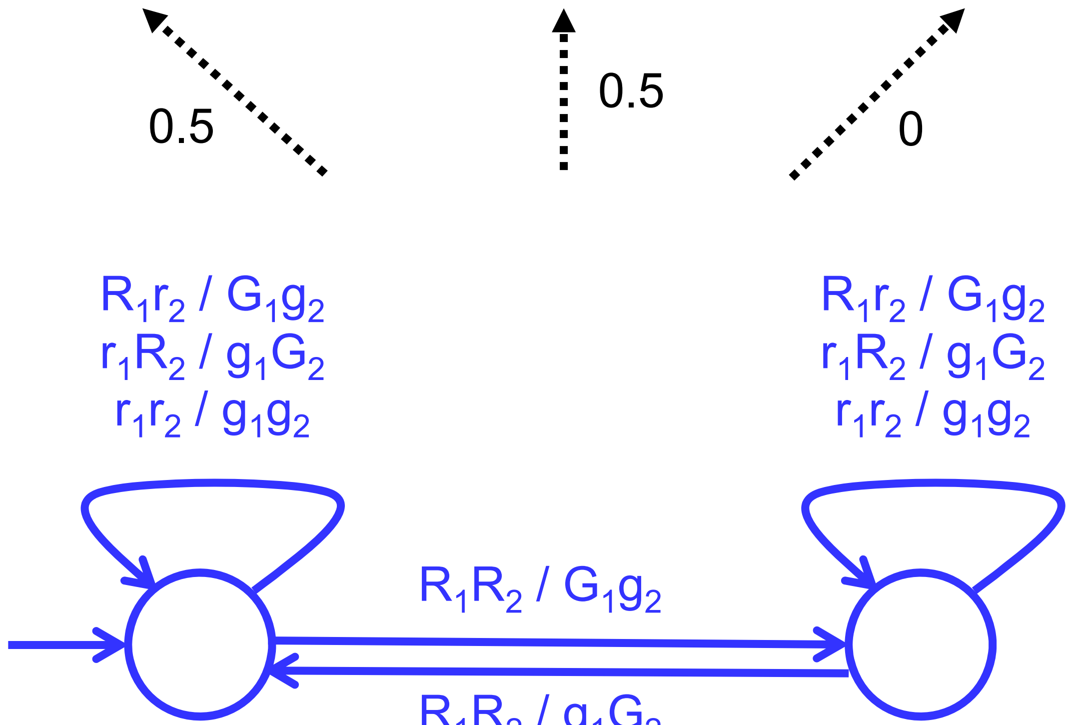


Example: Incompatible Requirements

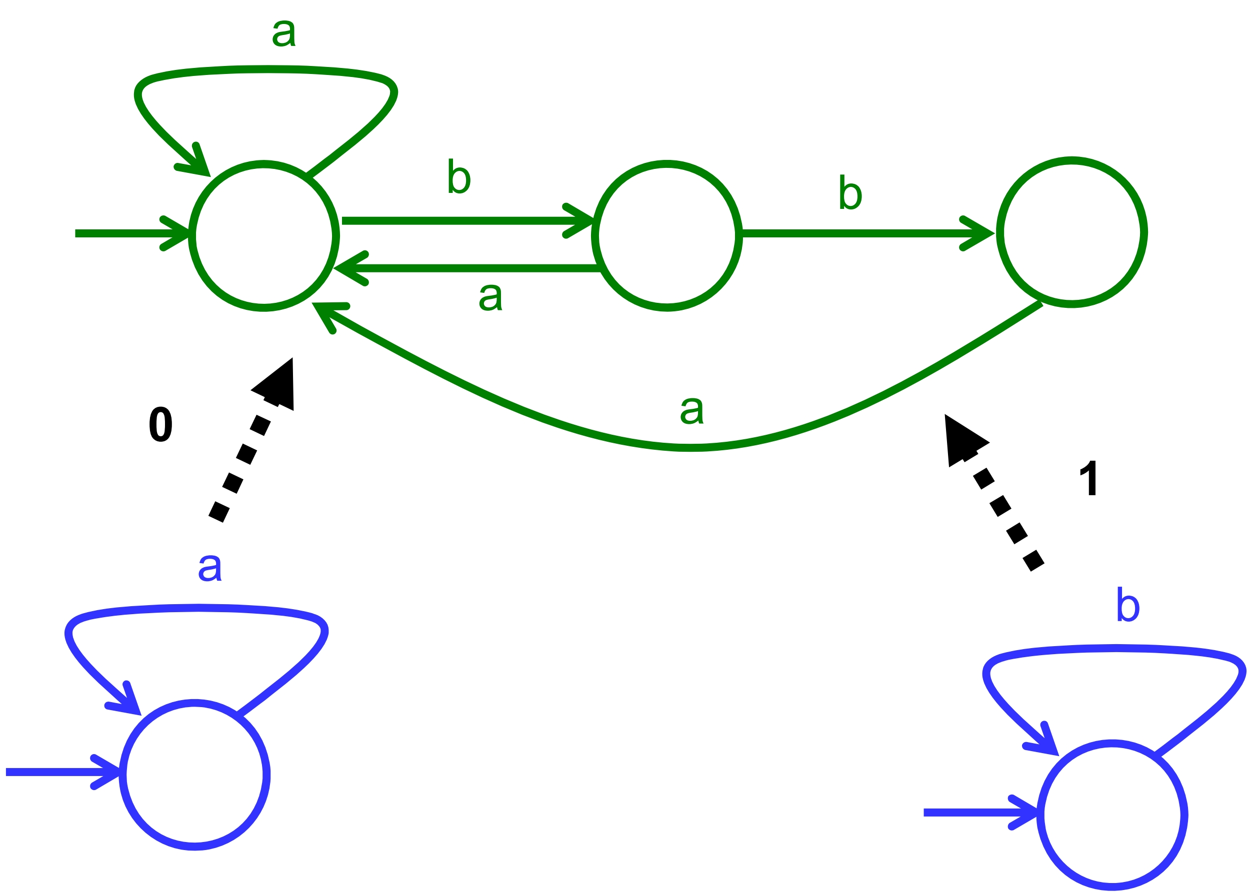


Example: Incompatible Requirements

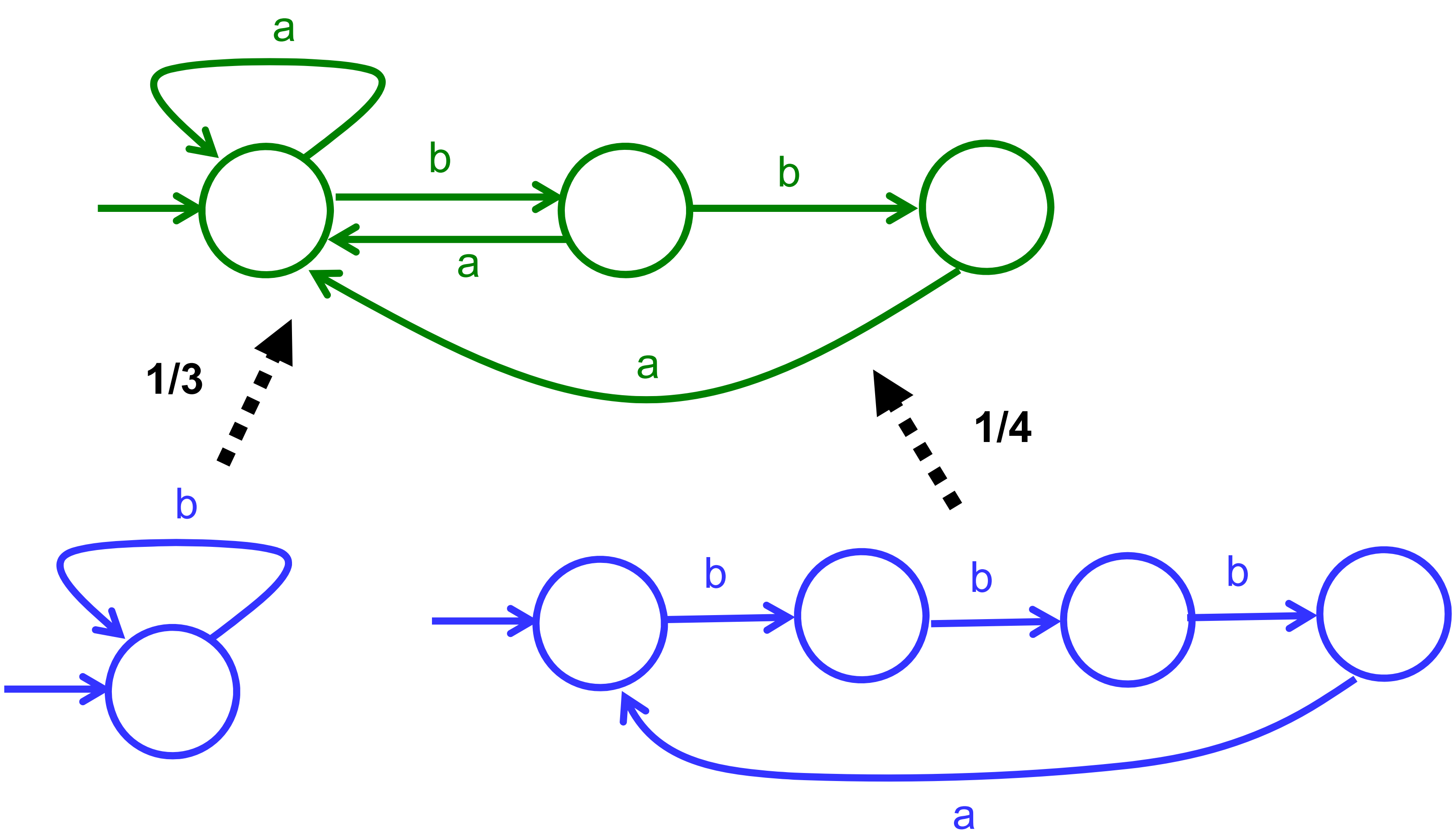
□ $(R_1 \;,\; G_1)$ □ $(R_2 \;,\; G_2)$ □ $= (G_1 - G_2)$



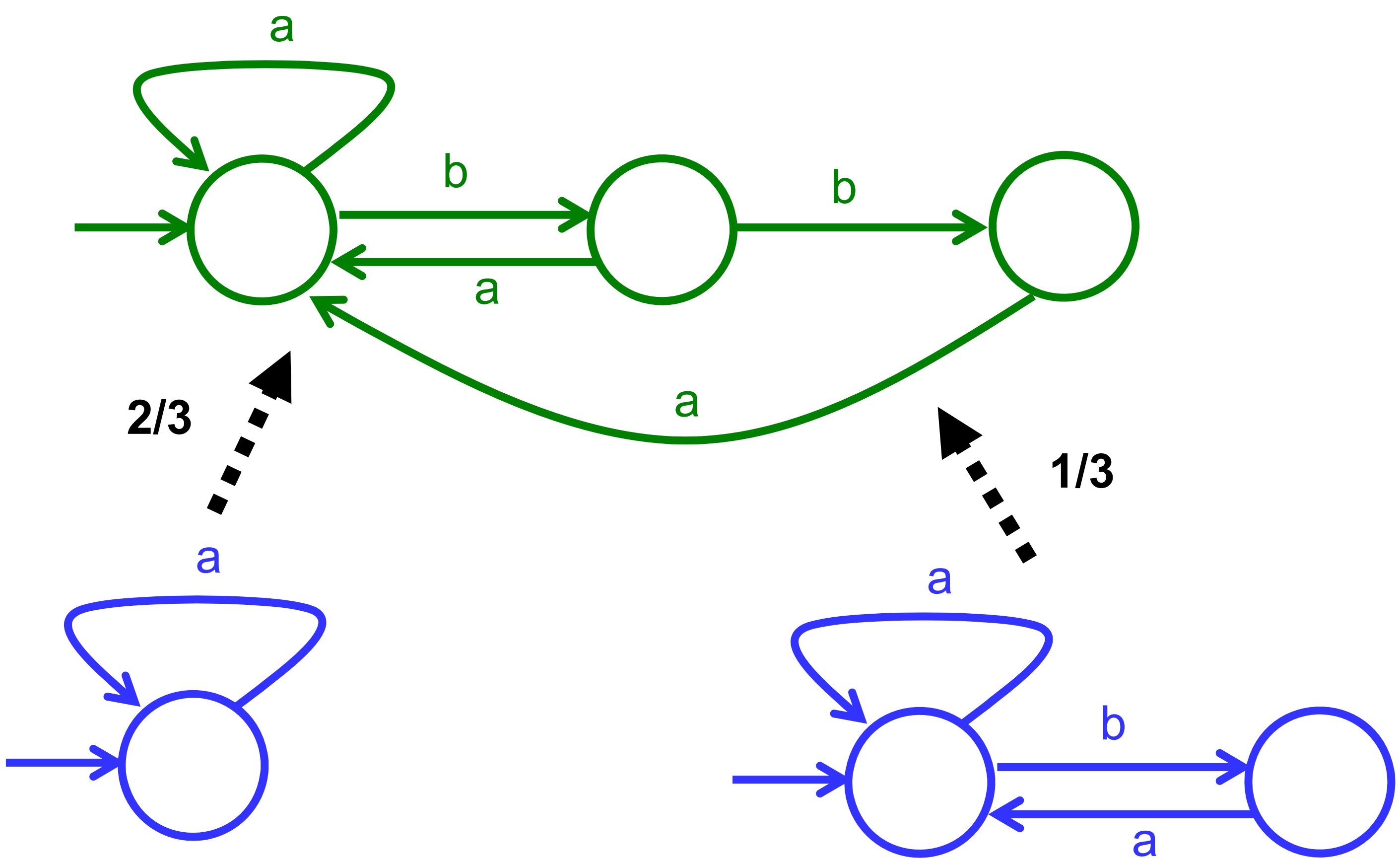
Simulation Preorder



Simulation Distance



Robustness Distance

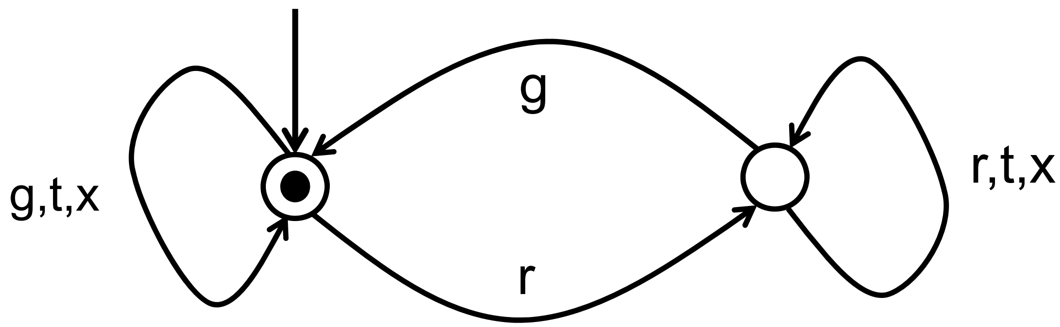


Response Times

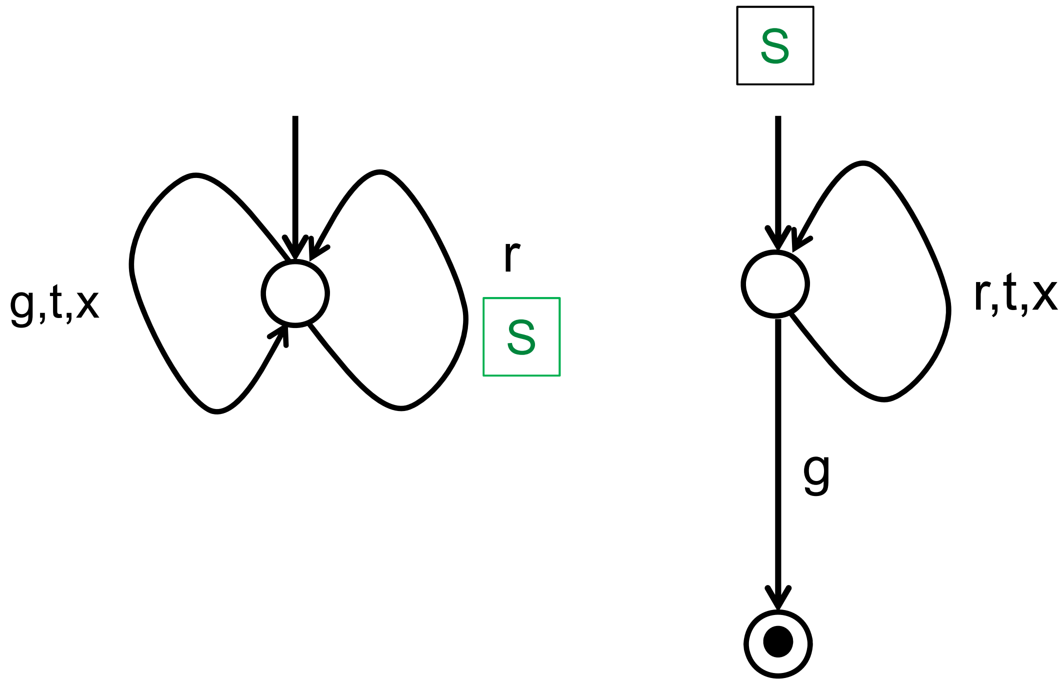
x t t x x r x x t x x x r x t x t x x t g r t t g g x t x t x … 4,3,2

r t t t t t t t … 4

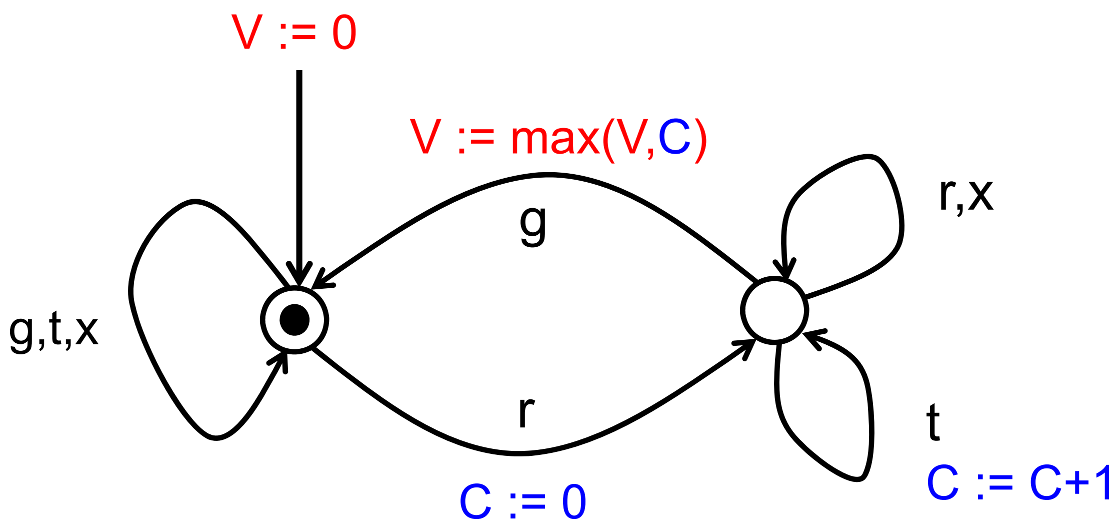
Response Property



Response Monitor

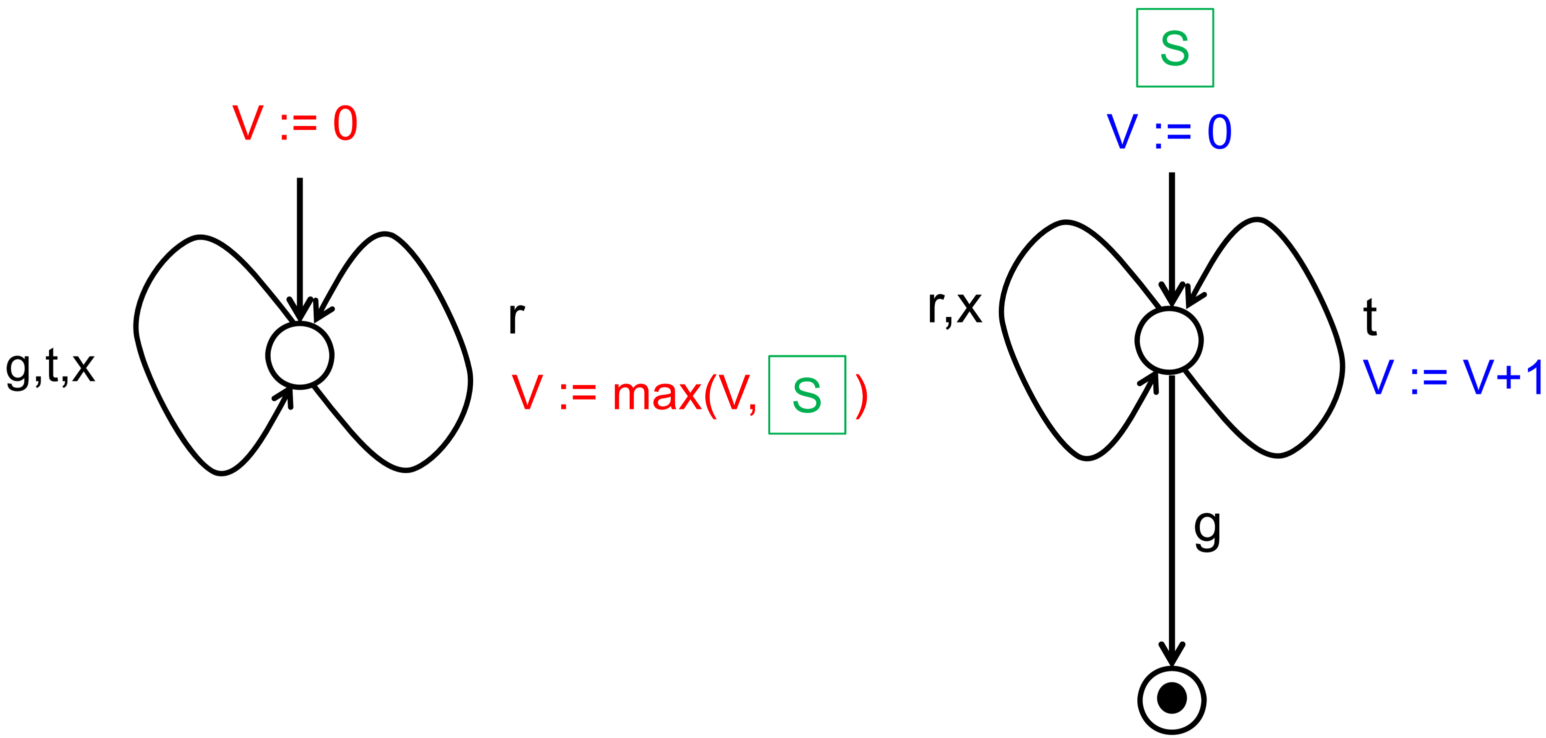


Maximal Response



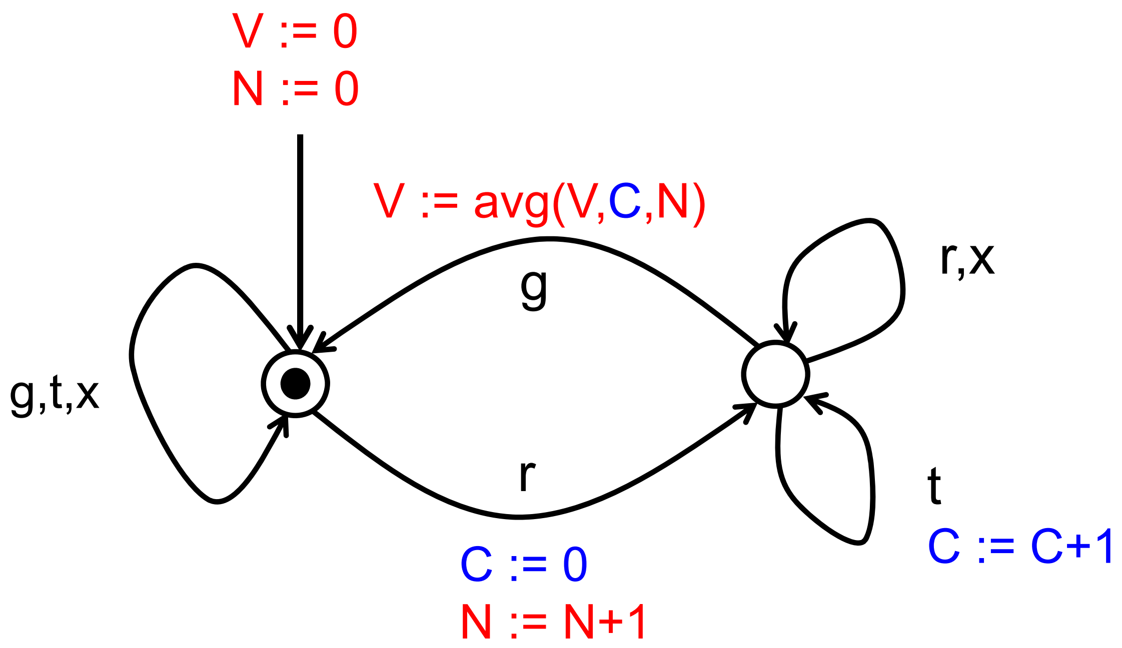
Value of an infinite run is liminf of V.

Maximal Response Monitor



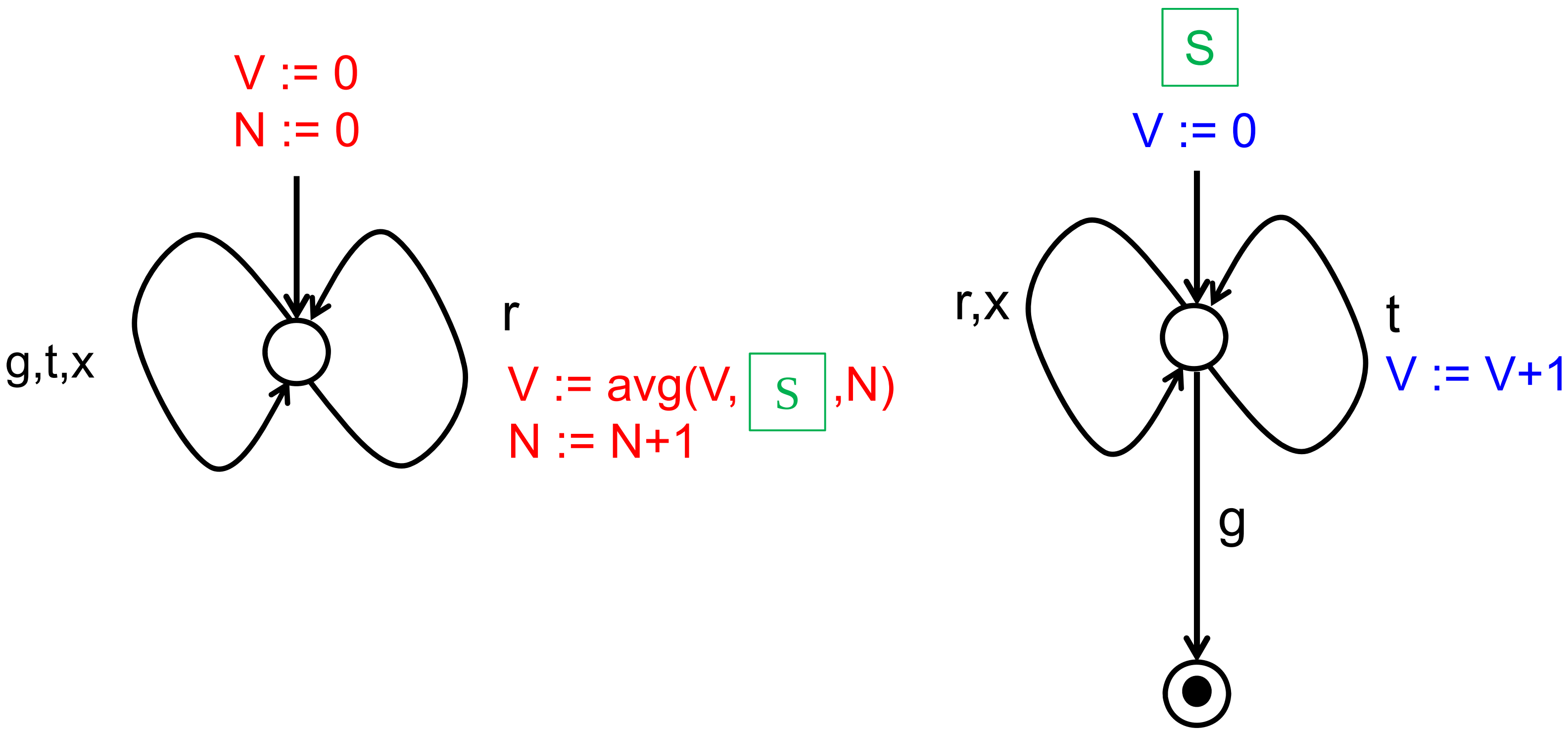
$\boxed{S}$ is final value of V.

Average Response

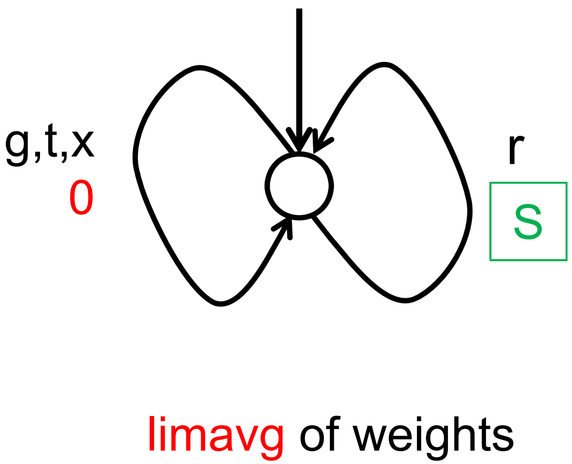


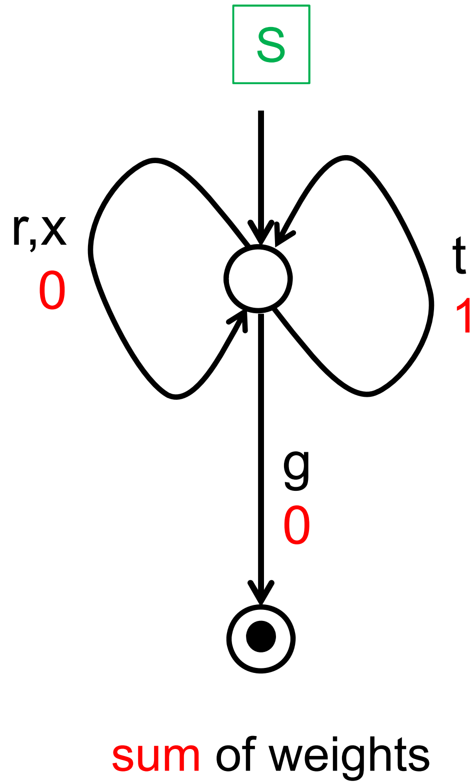
avg(V,C,N) = (V∗(N-1)+C) / N

Average Response Monitor



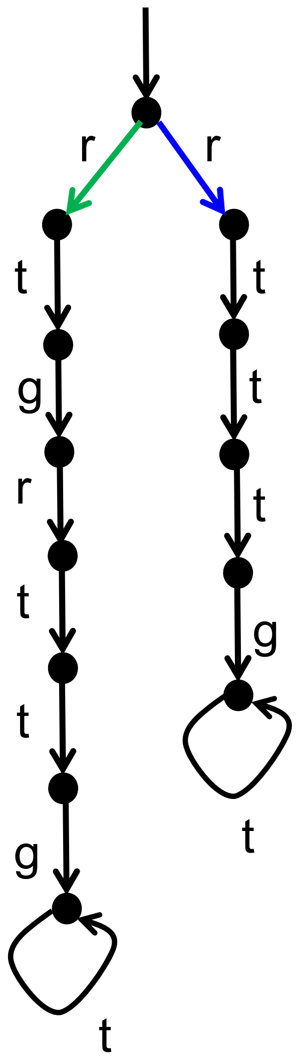
Nested Weighted Automata





Function on weights instead of registers.

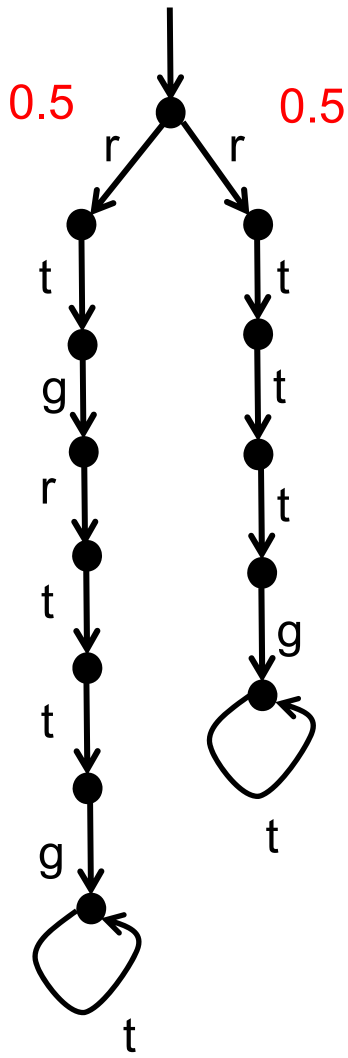
Example



Best maximal response time: 2
Worst maximal response time: 3
Emptiness/universality of (max,inc) automata.

Best average response time: 1.5
Worst average response time: 3
Emptiness/universality of (avg,inc) automata.

Probabilistic Example



Expected maximal response time: 2.5
Expectation of (max,inc) automata.

Expected average response time: 2.25
Expectation of (avg,inc) automata.

Results on (avg,inc) Automata

	Nondeterministic	Deterministic
Emptiness	[PSPACE,EXPSPACE]	[PSPACE,EXPSPACE]
Universality	undecidable	[PSPACE,EXPSPACE]
Expectation	?	PTIME

Quantitative Analysis Challenge:

Find the exact complexity of computing average response time over graphs.

Summary

There are plenty of reactive modeling challenges left,
not to speak of practical analysis methods and tools.

Some References

1 Simulation distances
[CONCUR 2010]

2 Nested weighted automata
[LICS 2015, 2016]

3 Co-synthesis
[TACAS 2007]

4 Multiple viewpoints
[EMSOFT 2008]