Modular Generic Verification of LTL Properties for Aspects

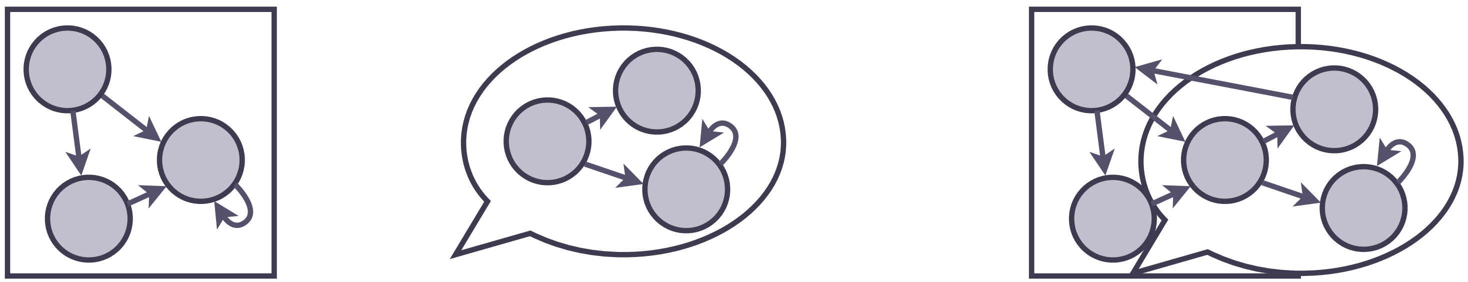
Shmuel Katz
Max Goldman

{katz, mgoldman}@cs.technion.ac.il

FOAL '06

Aspects

- Base + Aspect = Augmented

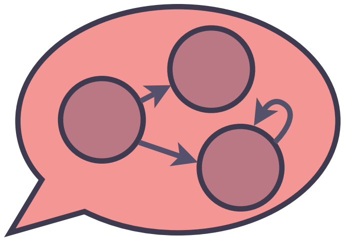
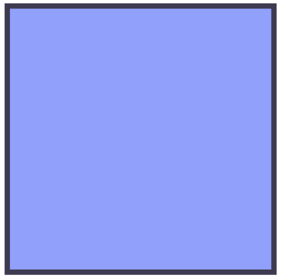


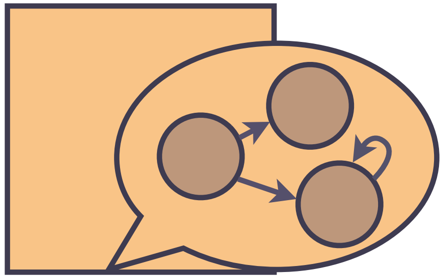
- State Machines
- Model Checking

Aspect Verification

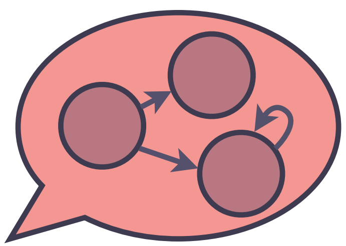
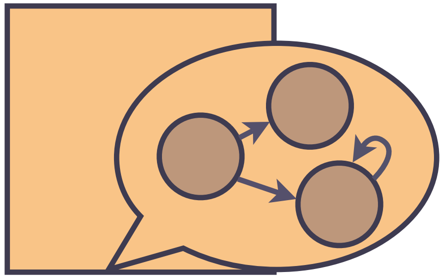
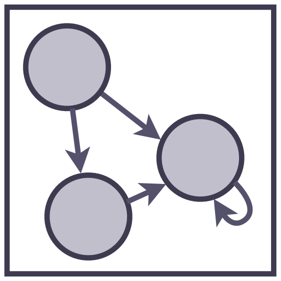
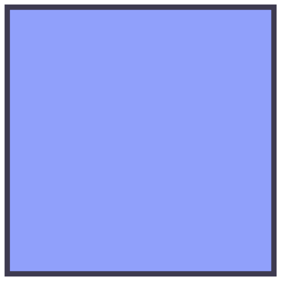
- Aspects have a specification
  - Requirements about base system
  - Results to hold in augmented system
- Prove once-and-for-all that an aspect satisfies its specification

Aspect Verification

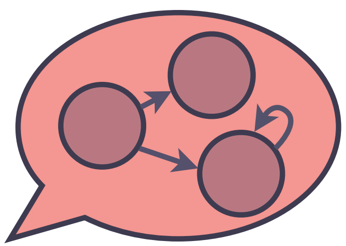
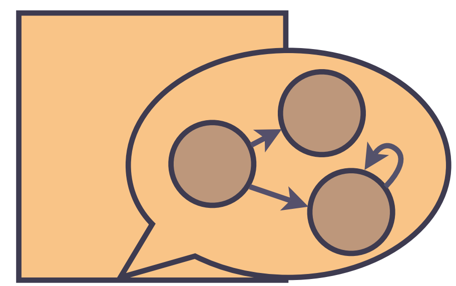
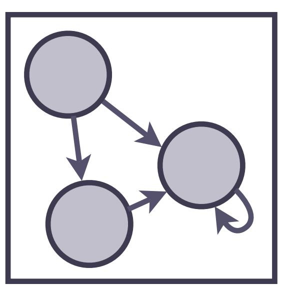
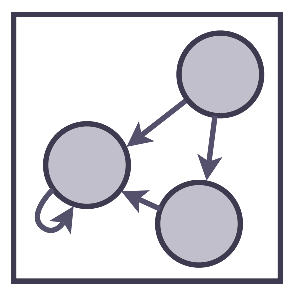
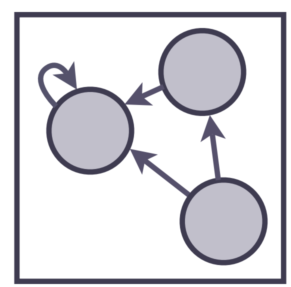
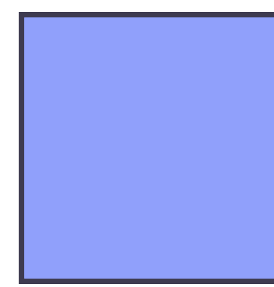
- Aspect  requires 

  and guarantees 

Modular

- Consider the aspect independently from the base machine
- Prove  guarantees 
- Prove  is 

Generic

- Consider the aspect independently from **any** base machine
- Prove  guarantees 
- Prove  or  or  is 

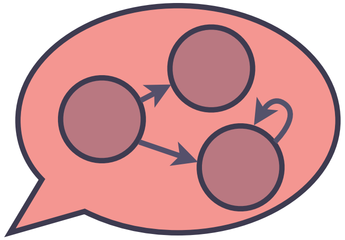
Idea

Aspect

- Advice: state machine A
- Pointcut: descriptor ρ
- Specification:
  - Base machine requirement ψ
  - Woven machine result φ

Aspect

- Advice: state machine

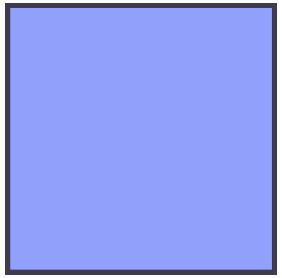


- Pointcut: descriptor ρ
- Specification:
  - Base machine requirement ψ
  - Woven machine result φ

Aspect

- Advice: state machine A
- Pointcut: descriptor ρ
- Specification:
  - Base machine requirement ψ
  - Woven machine result φ

Aspect

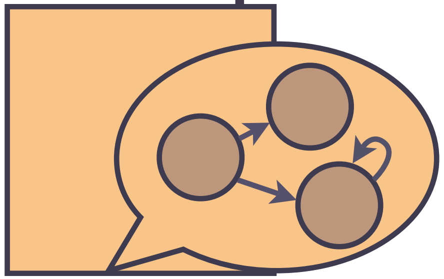
- Advice: state machine A
- Pointcut: descriptor ρ
- Specification:
  - Base machine requirement 
  - Woven machine result φ

Aspect

- Advice: state machine A
- Pointcut: descriptor ρ
- Specification:
  - Base machine requirement ψ
  - Woven machine result φ

Aspect

- Advice: state machine A
- Pointcut: descriptor ρ
- Specification:
  - Base machine requirement ψ
  - Woven machine result



Aspect

- Advice: state machine A
- Pointcut: descriptor ρ
- Specification:
  - Base machine requirement ψ
  - Woven machine result φ

Goal

- Prove
  - For all base machines B
  - If “B satisfies ψ”
  - Then “B woven with A according to ρ satisfies φ”

Problem

- What if the aspect puts the base program into a state it could never reach on its own?
- The behavior of the base program is unknown

Weakly Invasive

- Aspect returns to the base program only in states reachable by that base program on its own

  - Spectative
  - Regulative
  - Invasive within original domain

Result

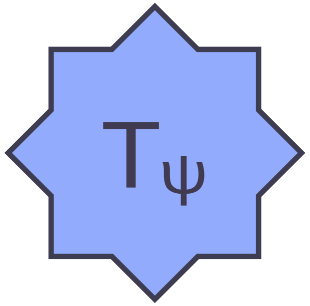
- Prove
  - For all base machines B
  - If "B satisfies ψ"
  - And "A with ρ is weakly invasive for B"
  - Then "B woven with A according to ρ satisfies φ"

Strategy

- Build a "generic" state machine version of assumption ψ
- Weave the aspect into this model
- Prove that this augmented generic model satisfies the desired result

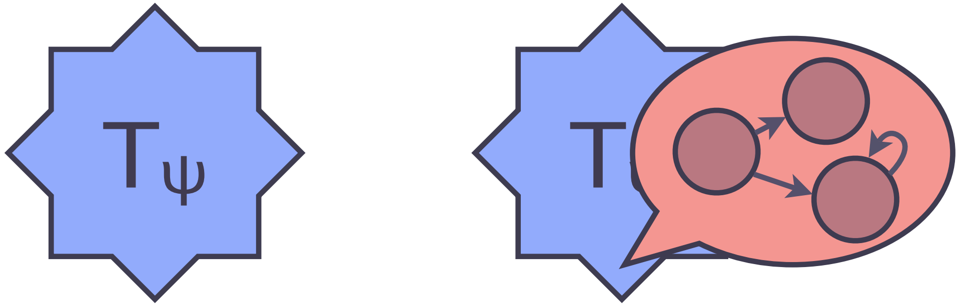
Strategy

- Build a “generic” state machine version of assumption ψ
- Weave the aspect into this model
- Prove that this augmented generic model satisfies the desired result



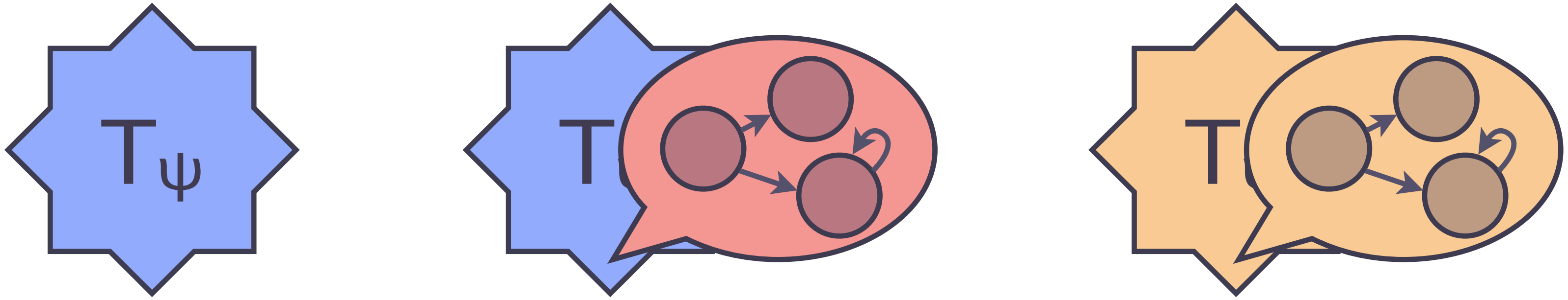
Strategy

- Build a “generic” state machine version of assumption ψ
- Weave the aspect into this model
- Prove that this augmented generic model satisfies the desired result



Strategy

- Build a “generic” state machine version of assumption ψ
- Weave the aspect into this model
- Prove that this augmented generic model satisfies the desired result



Components

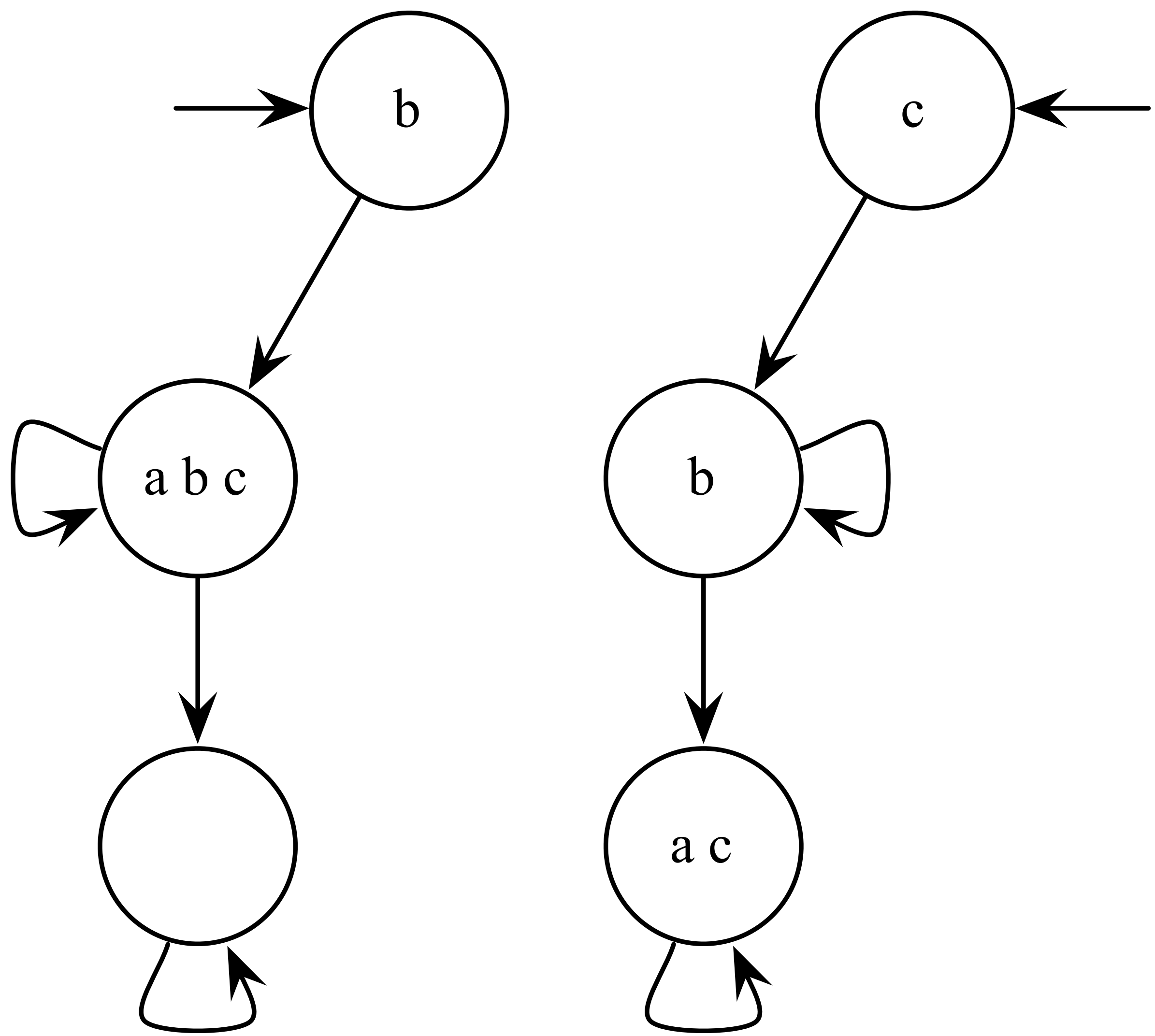
State Machines

- Finite set of states
- Set of atomic propositions
- Labels
- Nondeterminism

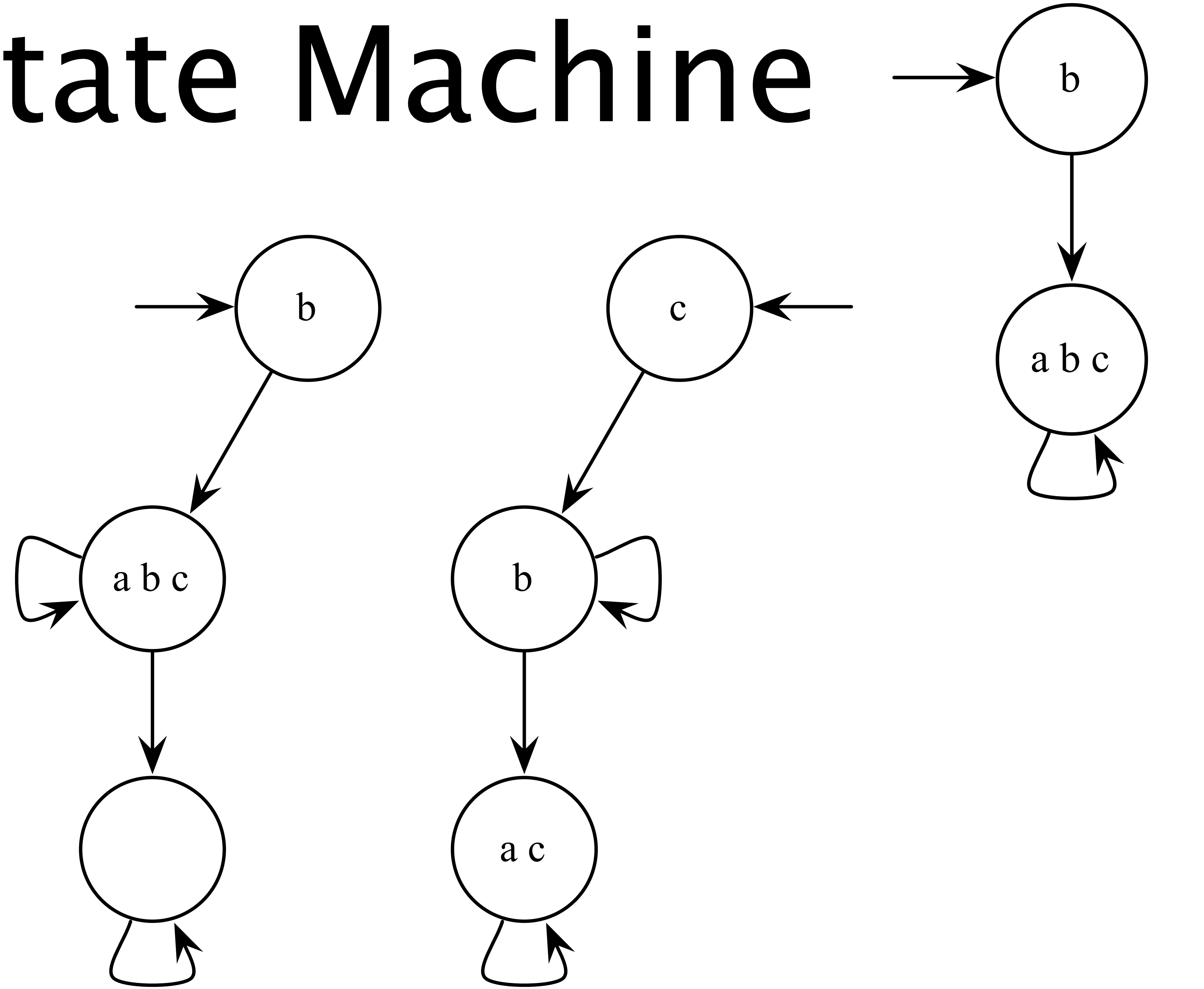
State Machines

- Finite set of states S
- Set of atomic propositions AP
- Labeling function $L : S \rightarrow 2^{AP}$
- Path relation R containing pairs (s,t) when there is a transition from s to t

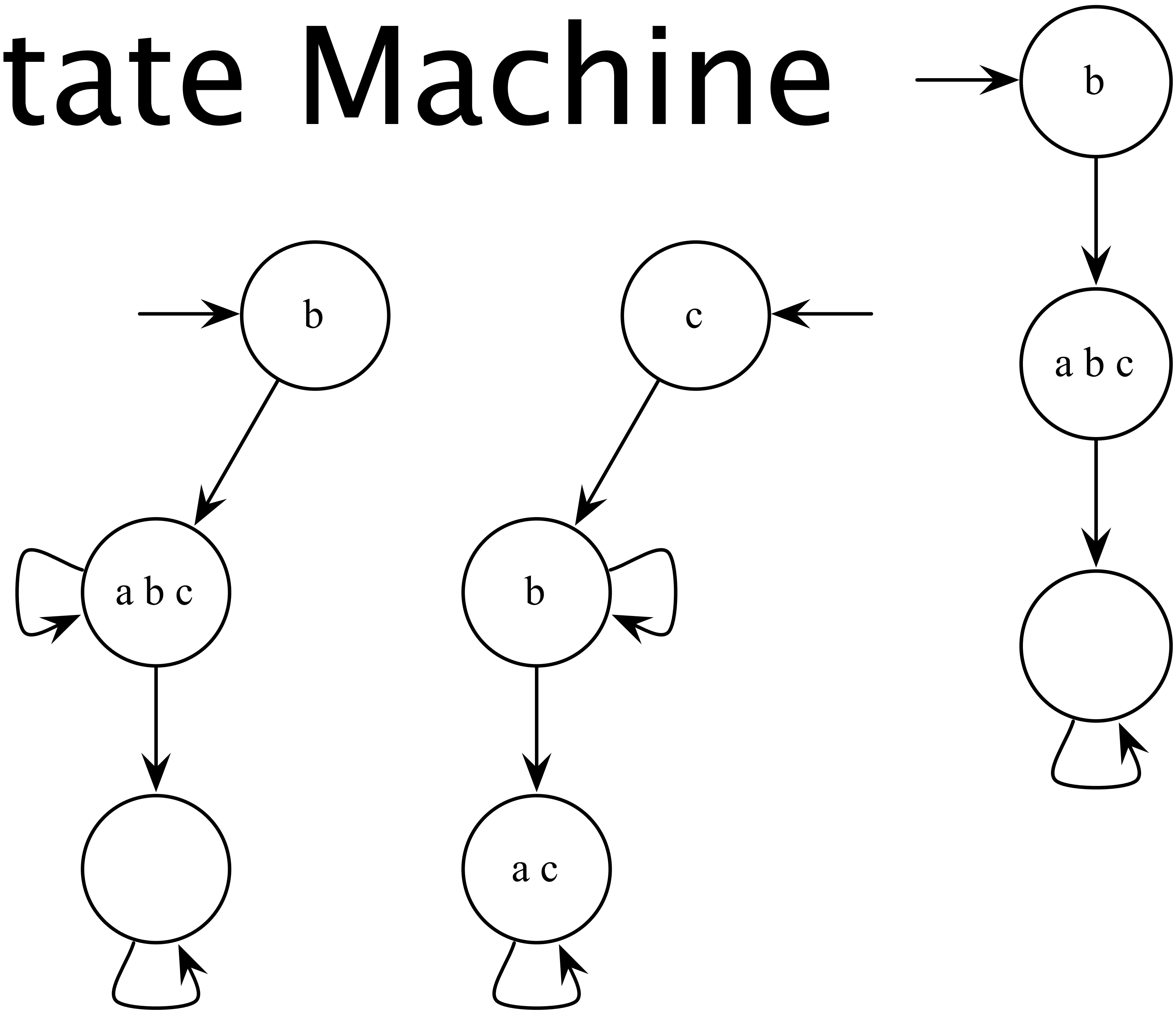
State Machine



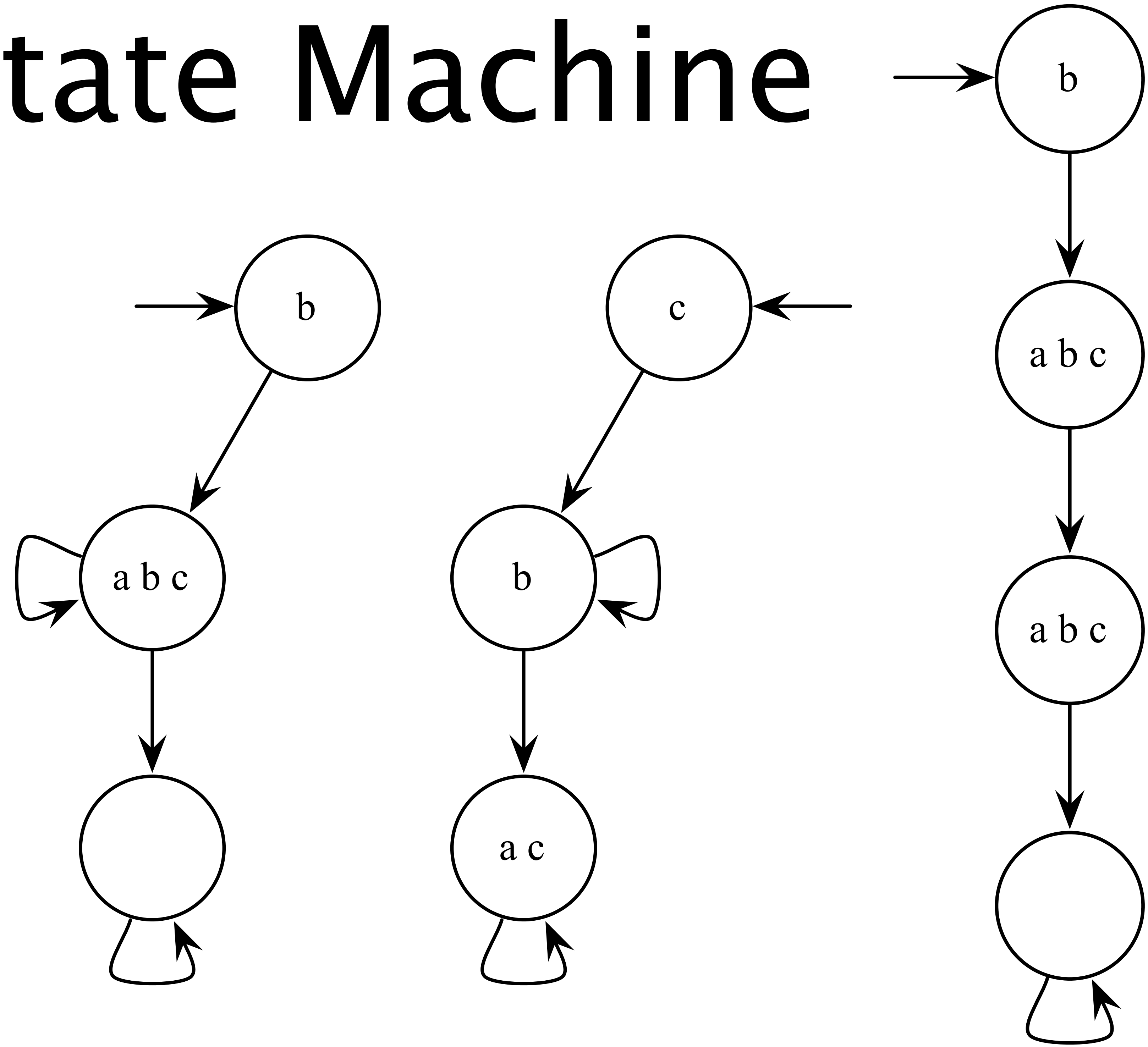
State Machine



State Machine



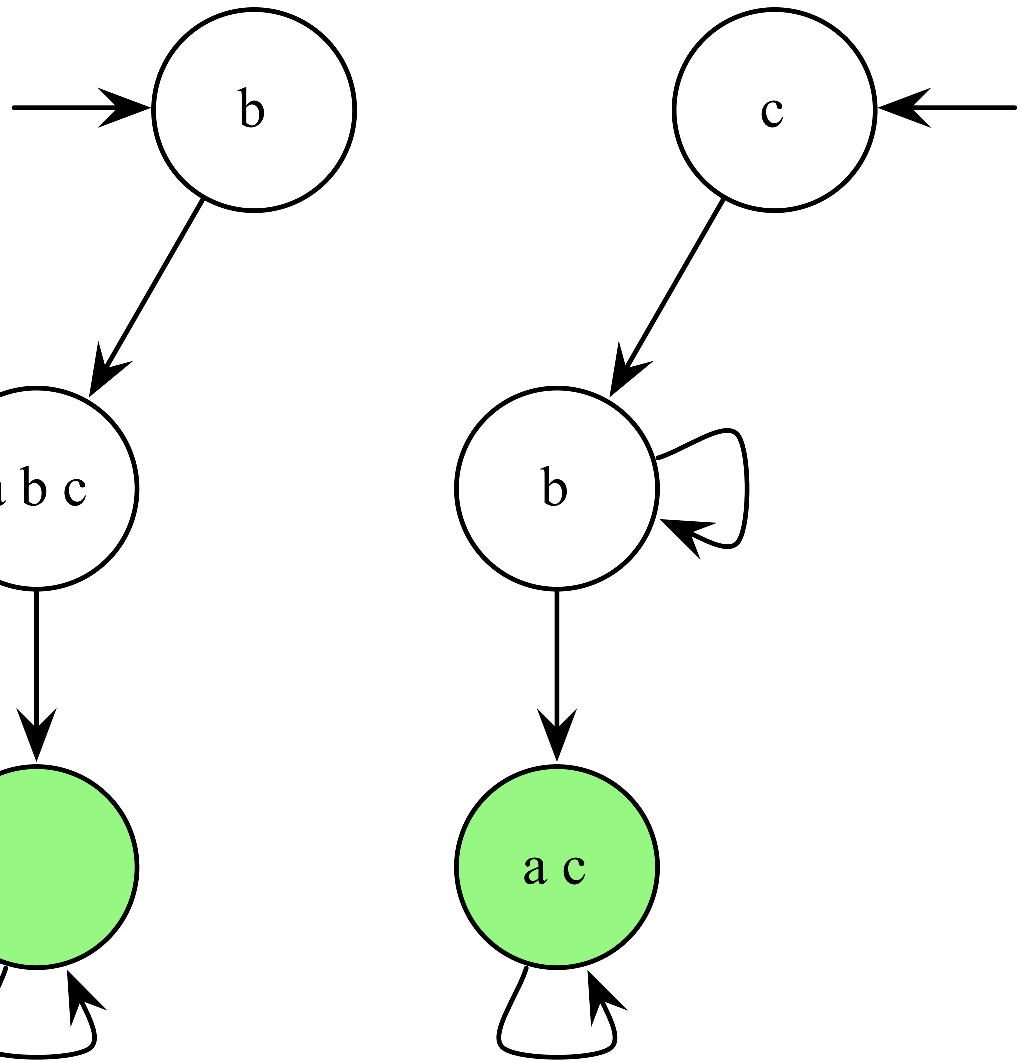
State Machine



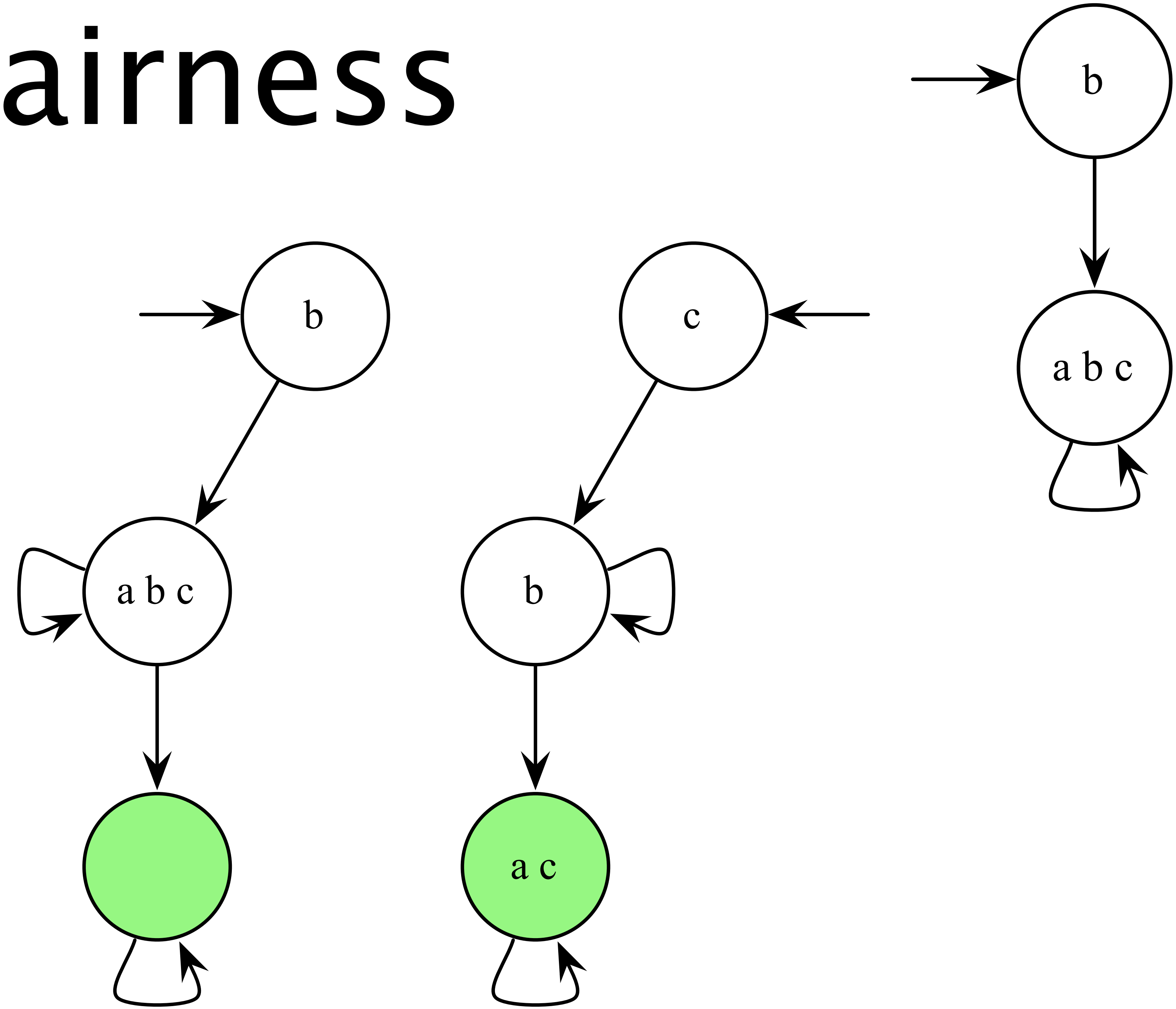
Fairness

- Problem with nondeterminism: often allows the system to “do nothing” forever
- Impose a fairness constraint, and only look at fair paths
- Fairness set F: set of subsets of S
  - A path is fair iff it visits every set in F infinitely often

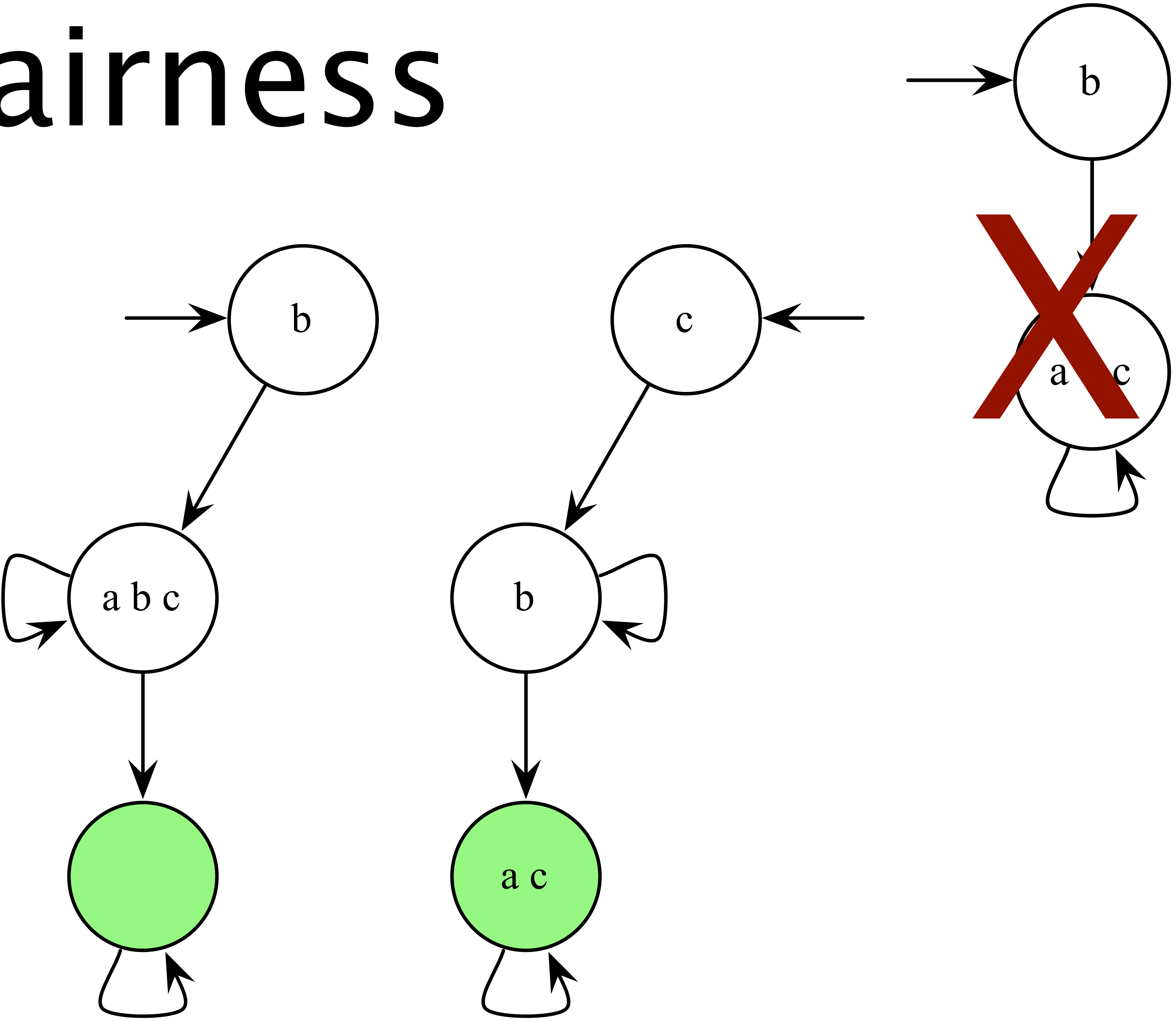
Fairness



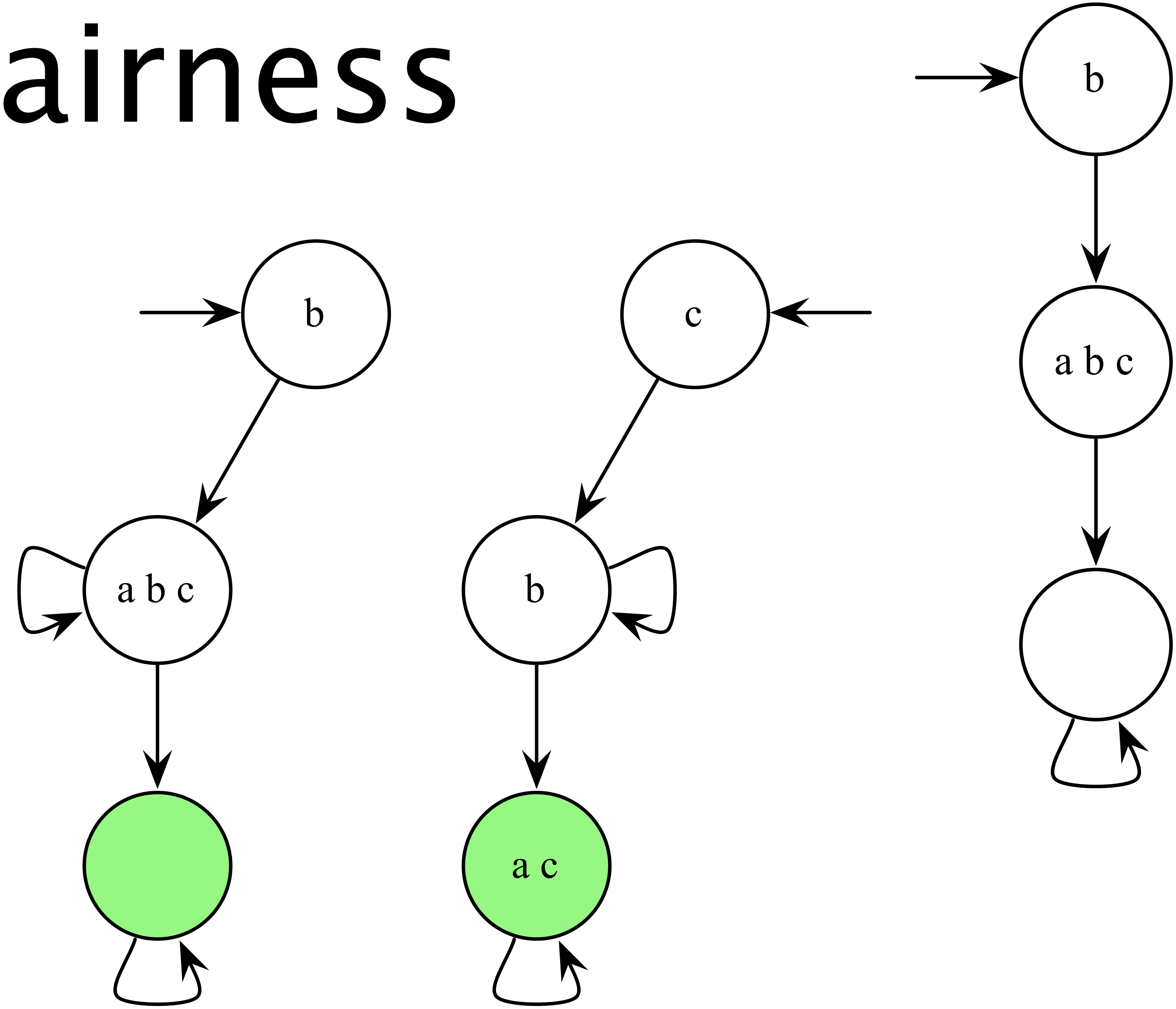
Fairness



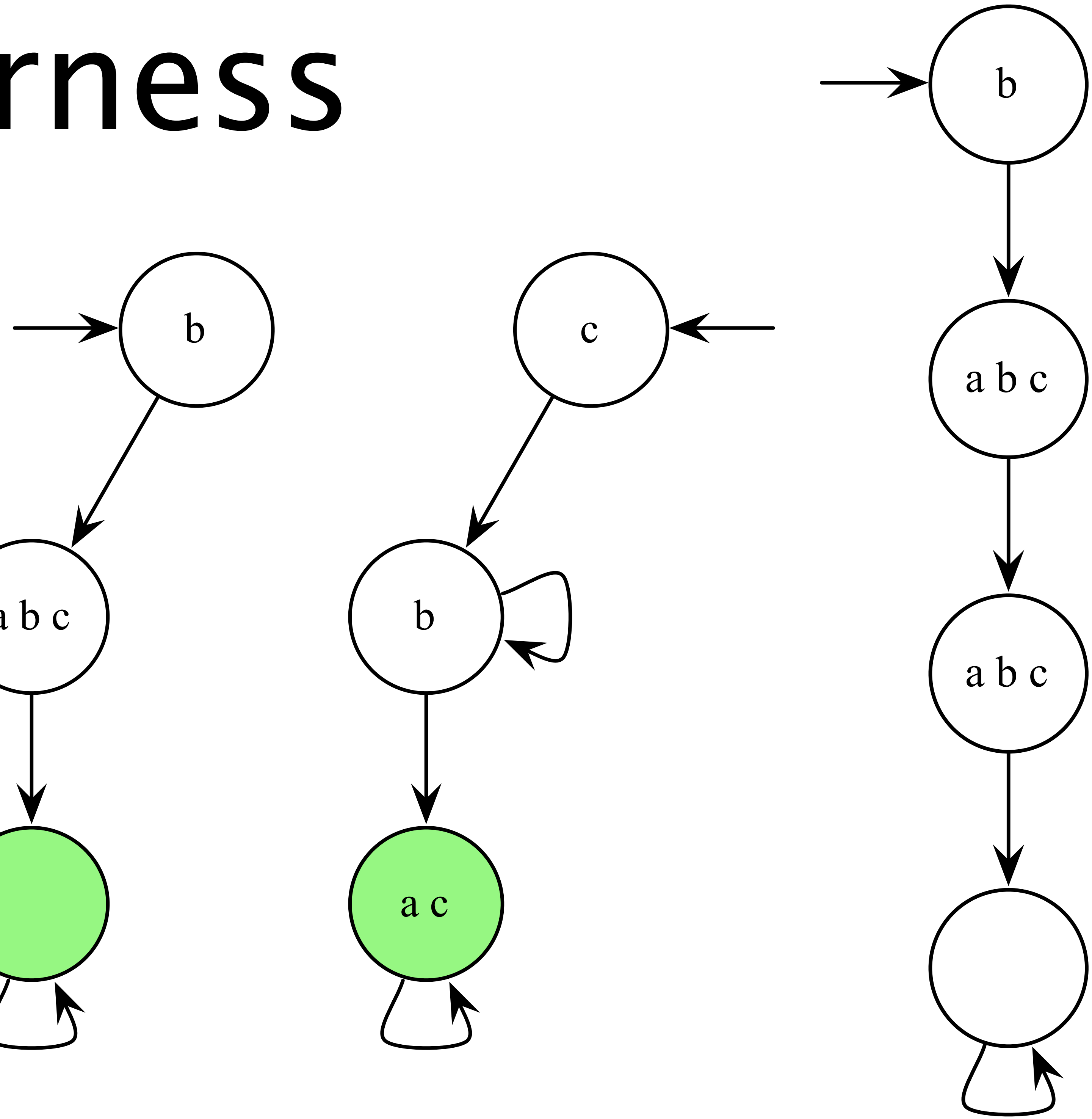
Fairness



Fairness

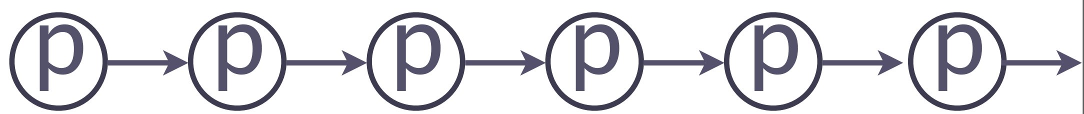


Fairness

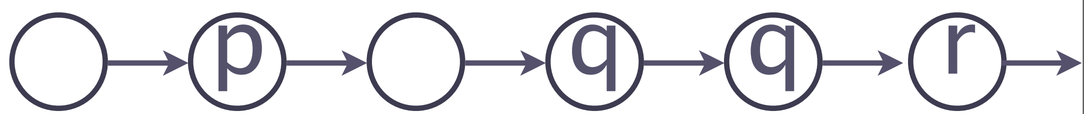


LTL

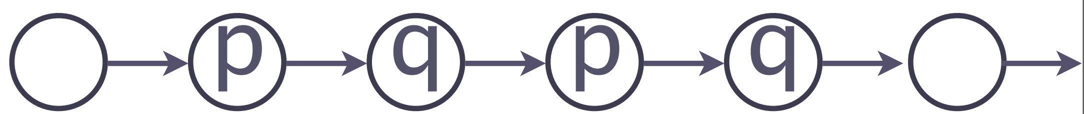
- Linear Temporal Logic
  - Logic of infinite paths of computation
- Path formulas
  - $G\ p$



  - $p \rightarrow F\ (q\ U\ r)$

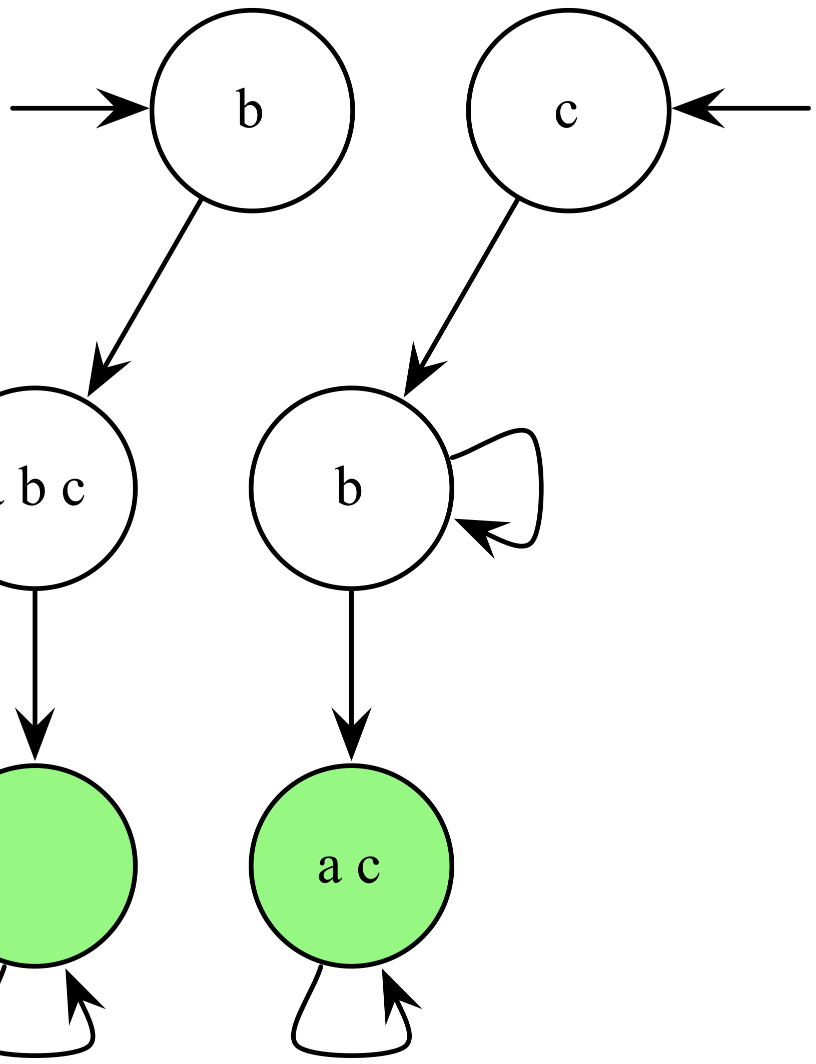


  - $p \rightarrow X\ q$



LTL Formulas

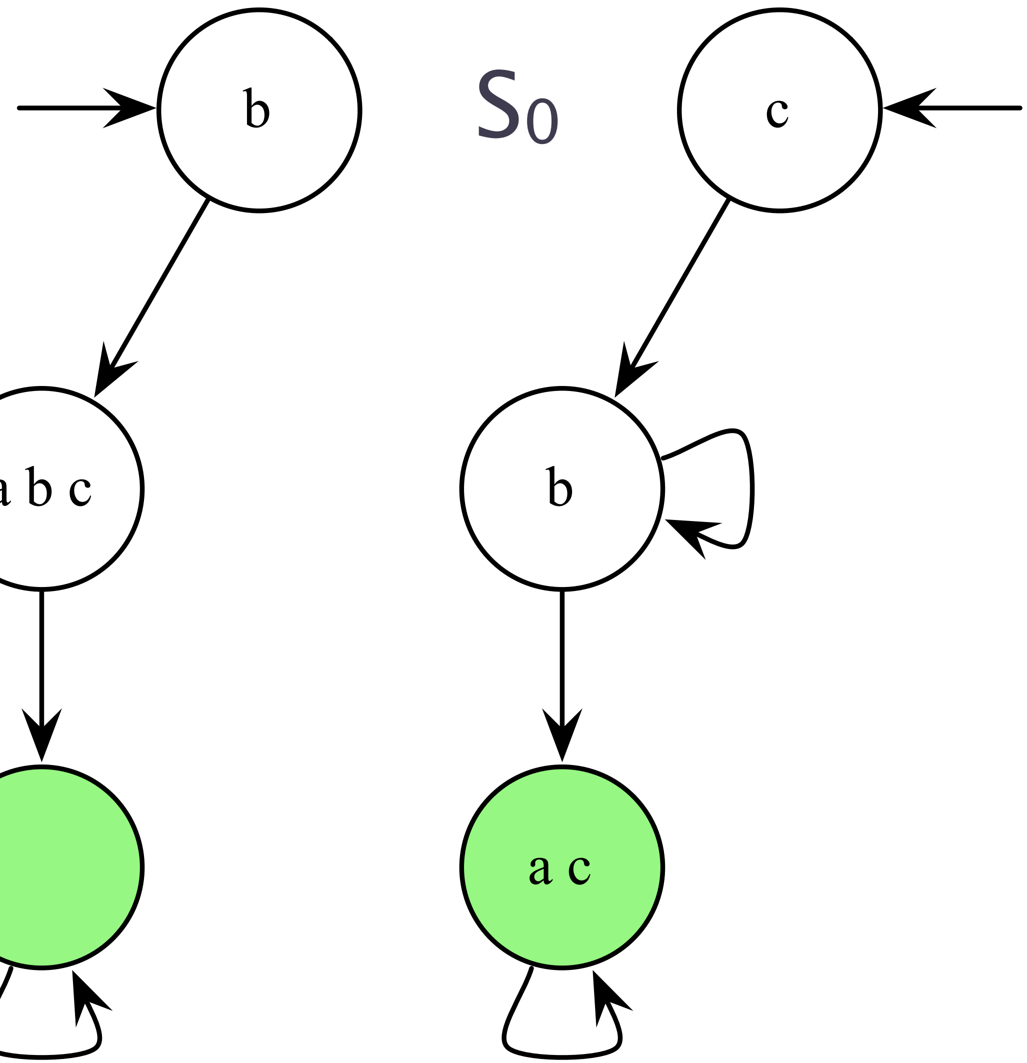
- A F c
- A F G ¬b
- A G ((¬a ∧ b) → F a)



Base Machine

- State machine B
  - Computation starts from one of the initial states $S_0 \subseteq S$

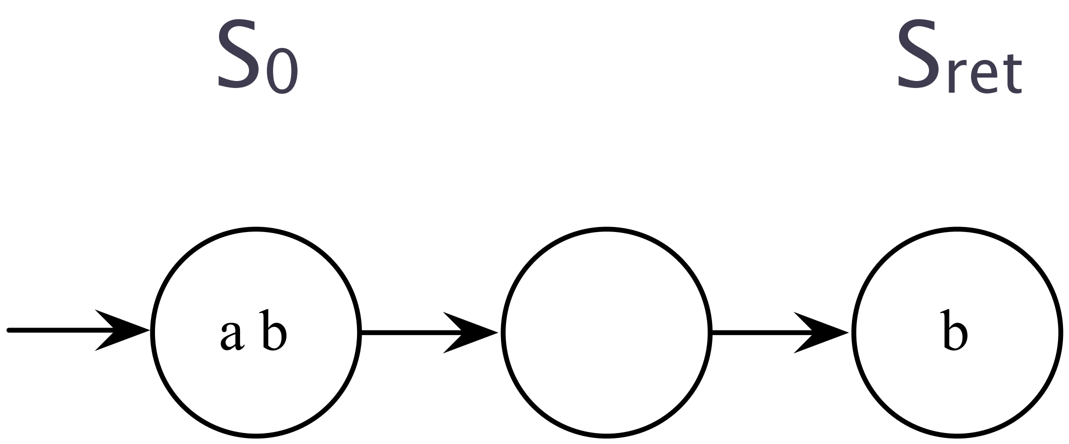
Base Machine



Advice

- State machine A
  - Initial states S_0
  - Return states S_{ret}

Advice

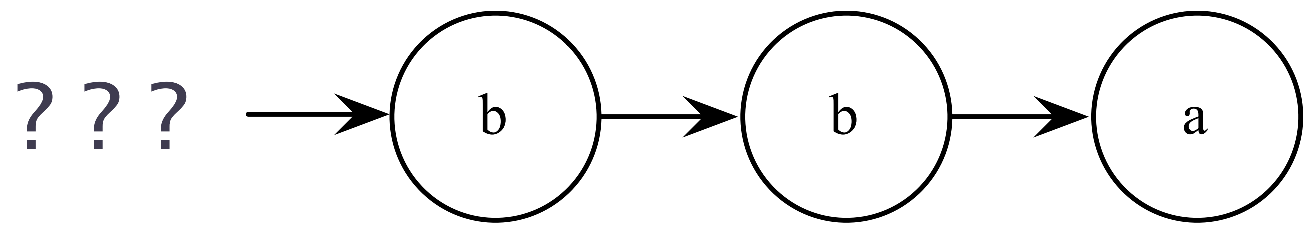


Pointcuts

- Pointcut descriptor ρ
  - Matches the end of a path
  - Past LTL, regular expressions, ...

Pointcut

$\rho = a \wedge Y\, b \wedge Y\, Y\, b$



Components

- State machines
- Fairness
- LTL
- Base machines
- Aspect advice machines
- Aspect pointcuts

Weaving

- Inputs:
  - Base machine B
  - Aspect machine A
  - Pointcut ρ
- Output:
  - Woven machine $\tilde{B}$

Weaving A with B

- Step 1: Make B pointcut-ready for ρ
  - Result: Machine B^{ρ}
- Step 2: Augment B^{ρ} with A
  - Result: Augmented machine $\tilde{B}$

1. Pointcut-Ready

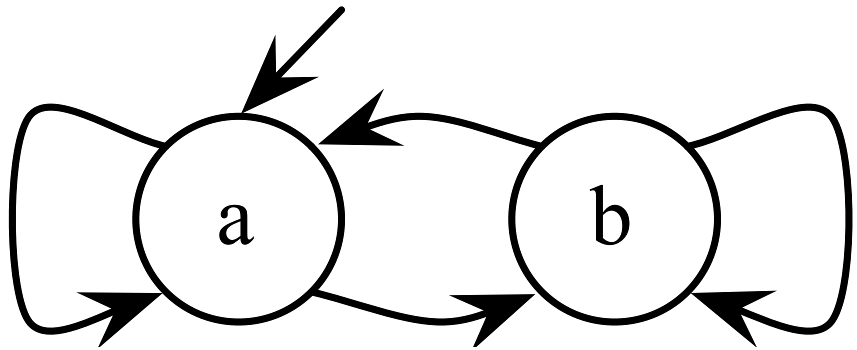
- Advantage: simplicity
- Disadvantage: static, not dynamic

- No problems for many aspects
  - State pointcut
  - Method call pointcut

1. Pointcut-Ready

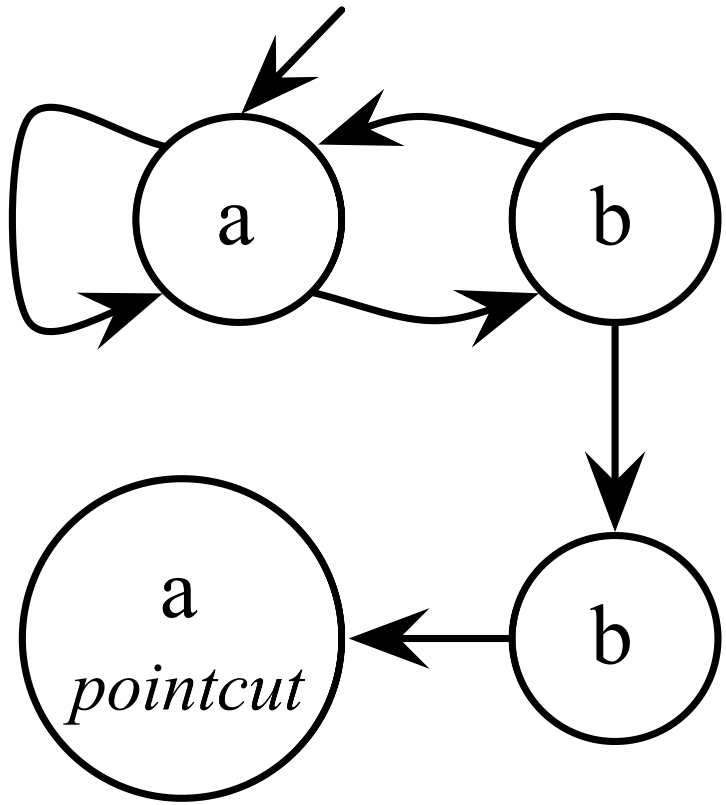
- Unwinding of paths such that each state either definitely does or definitely does not match the pointcut
- Matching states are labeled ‘pointcut’

1. Pointcut-Ready



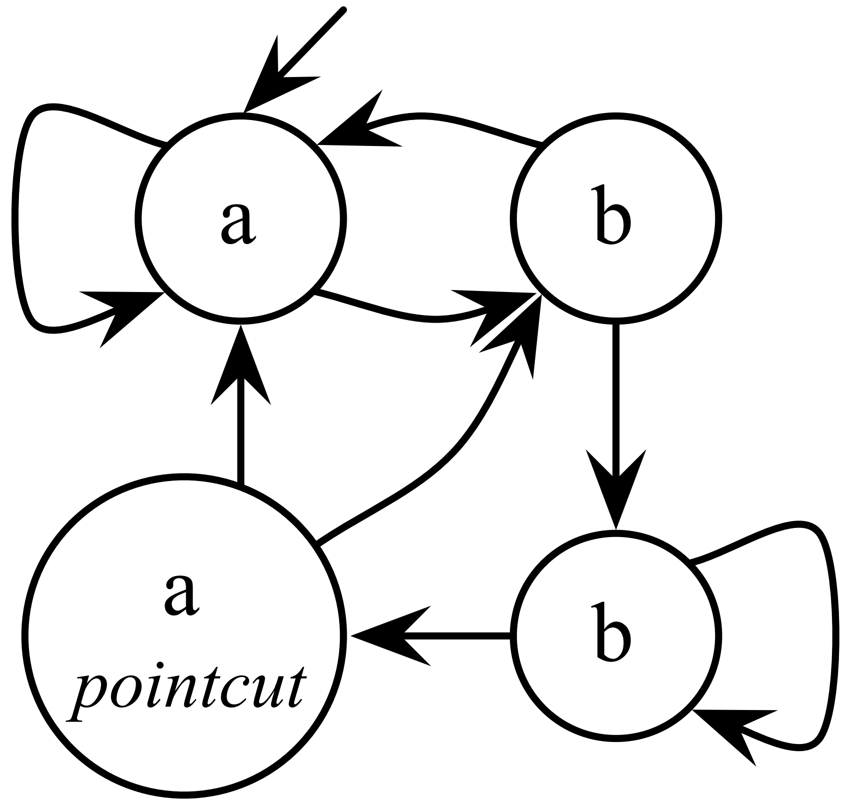
$\rho = a \wedge Y\, b \wedge Y\, Y\, b$

1. Pointcut-Ready



$\vdash \rho = a \wedge Y\, b \wedge Y\, Y\, b$

1. Pointcut-Ready



$\rho = a \wedge Y\, b \wedge Y\, Y\, b$

2. Augmented

- Transitions from base machine ‘pointcut’ states to aspect initial states
- Transitions from aspect return states to base machine states
- According to state labels

2. Augmented

- Rule: add all edges
  - ‘pointcut’ → aspect initial
  - aspect return → base
- Where the labels match

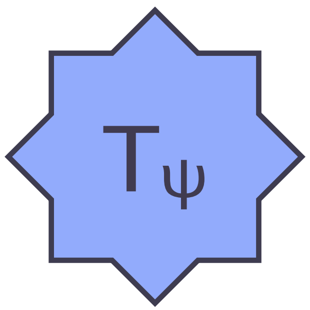
Weakly Invasive

- All edges from aspect return states go to reachable states in the base machine

Tableaux

Recall

A “generic” model built from the assumption formula ψ

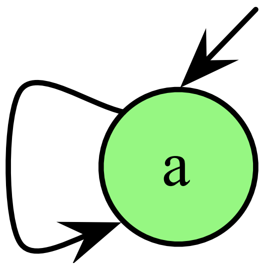


Tableaux

Exactly all the paths which satisfy a given LTL path formula

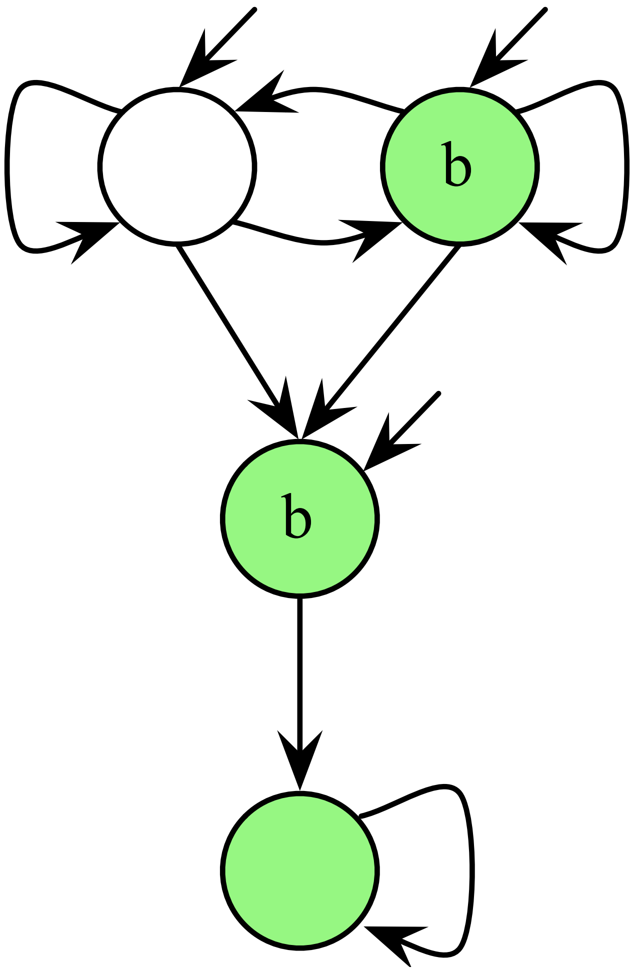
Tableau

G a



Tableau

$\vdash$ F b



Tableaux

- For a given LTL formula ψ
  - If a path supports the formula, it must be in the tableau
- For any machine satisfying ψ
  - All its paths must be in the tableau

Algorithm

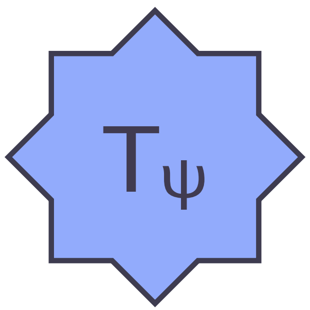
Recall

- Advice: state machine A
- Pointcut: descriptor ρ
- Specification:
  - Base machine requirement ψ
  - Woven machine result φ
- A, ρ, ψ, and φ over AP

Step 0

- Throw all the atomic propositions in AP into ψ, in clauses of the form
  - $\cdots \wedge (a \vee \neg a)$

Step 1

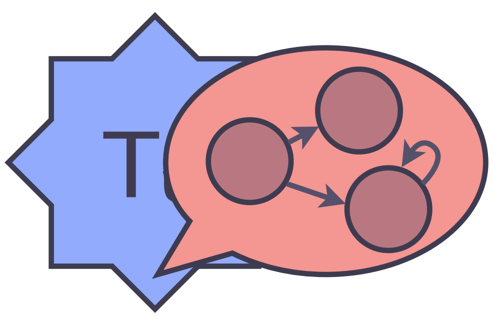
- Construct T_ψ, the tableau for ψ
- 

Step 2

- Restrict T_ψ to its reachable component

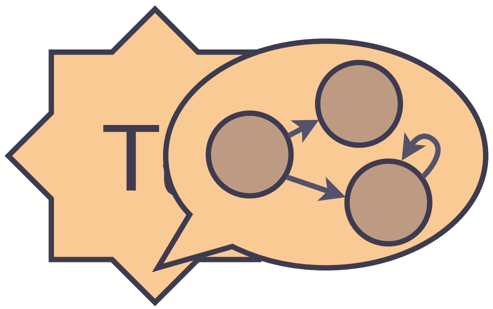
Step 3

- Weave A into T_ψ according to ρ
  - Result: $\widetilde{T_\psi}$



Step 4

- Determine if $\widetilde{T_\psi} \models \varphi$

- 

Claim

- If $\widetilde{T_\psi} \models \varphi$
- Then for any M
  - If $M \models \psi$
  - And A and ρ are weakly invasive for M
  - Then $\tilde{M} \models \varphi$

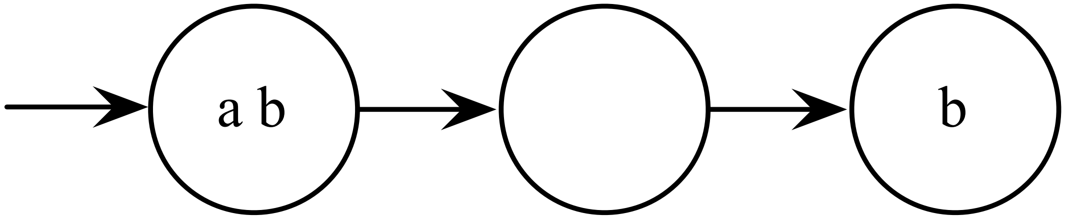
Proof

Outline

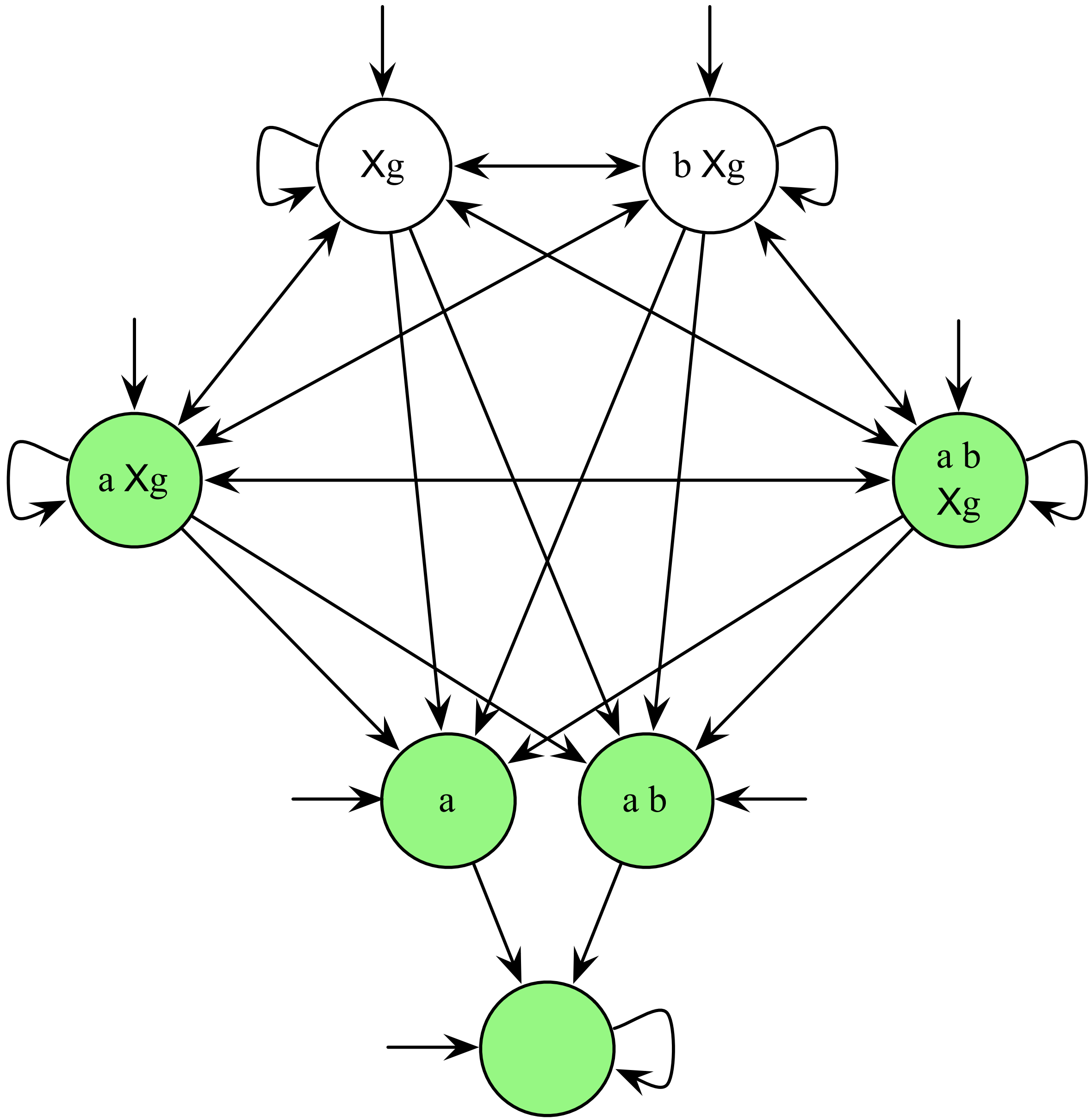
- T_ψ has every possible path
- So $\widetilde{T_\psi}$ has every possible augmented path
- If $\widetilde{T_\psi} \models \varphi$
- Then every possible augmented path supports φ

Example

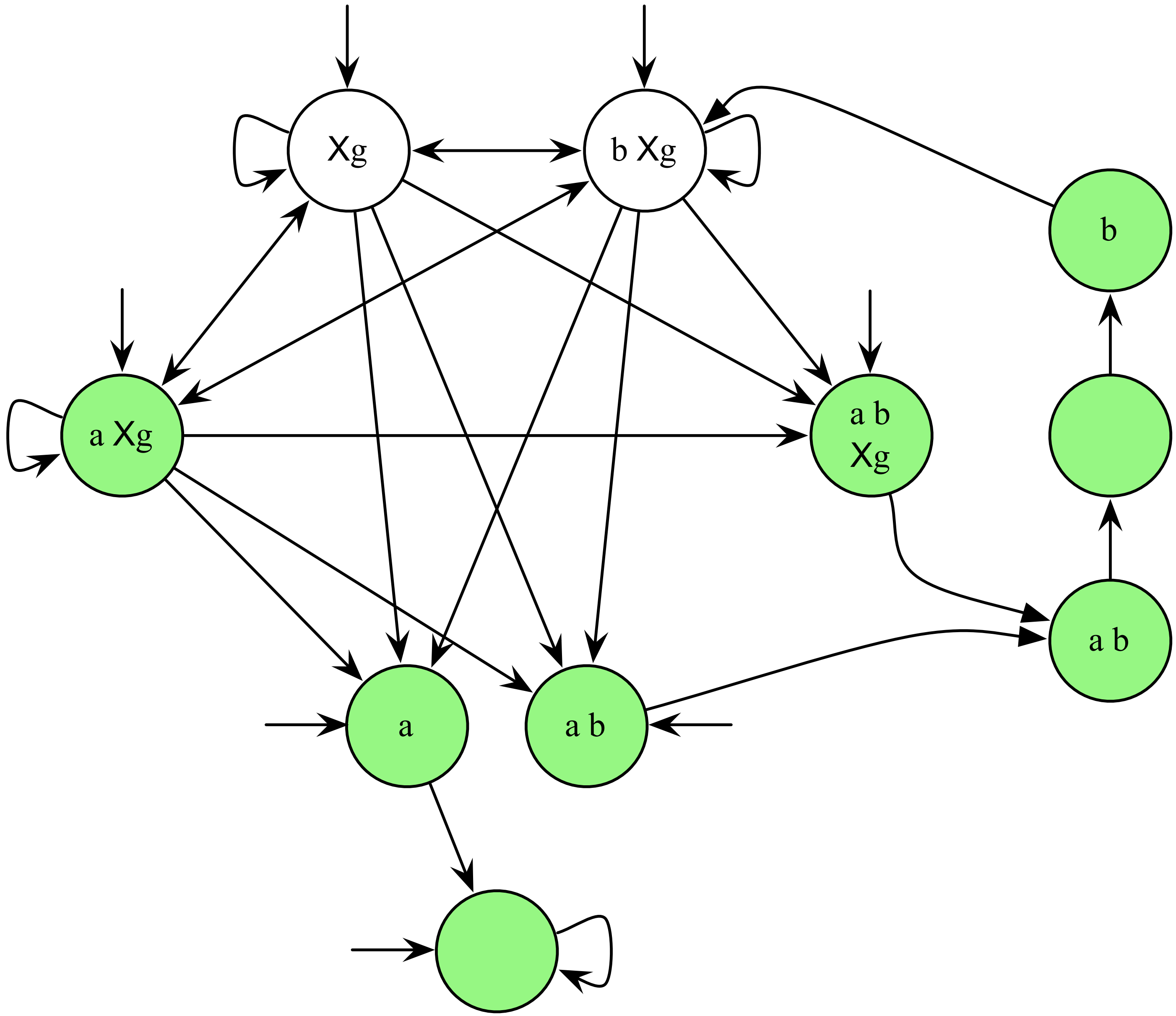
Aspect

- $\psi = \mathrm{A\,G}\,((\neg a \wedge b) \rightarrow \mathrm{F}\,a)$
- $\varphi = \mathrm{A\,G}\,((a \wedge b) \rightarrow \mathrm{X\,F}\,a)$
- $\rho = a \wedge b$
- $A =$ 

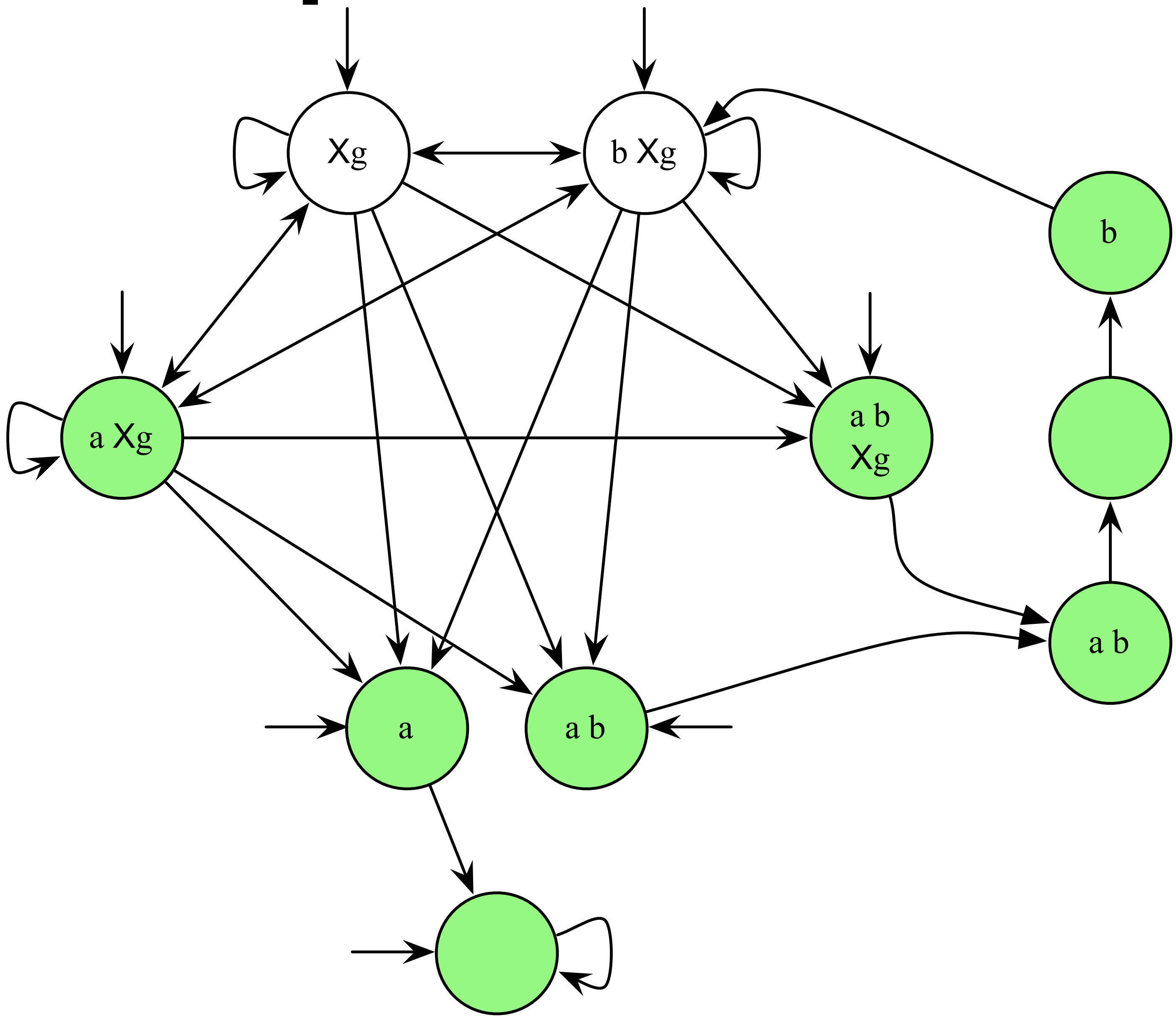
T_ψ



$\widetilde{T}_\psi$



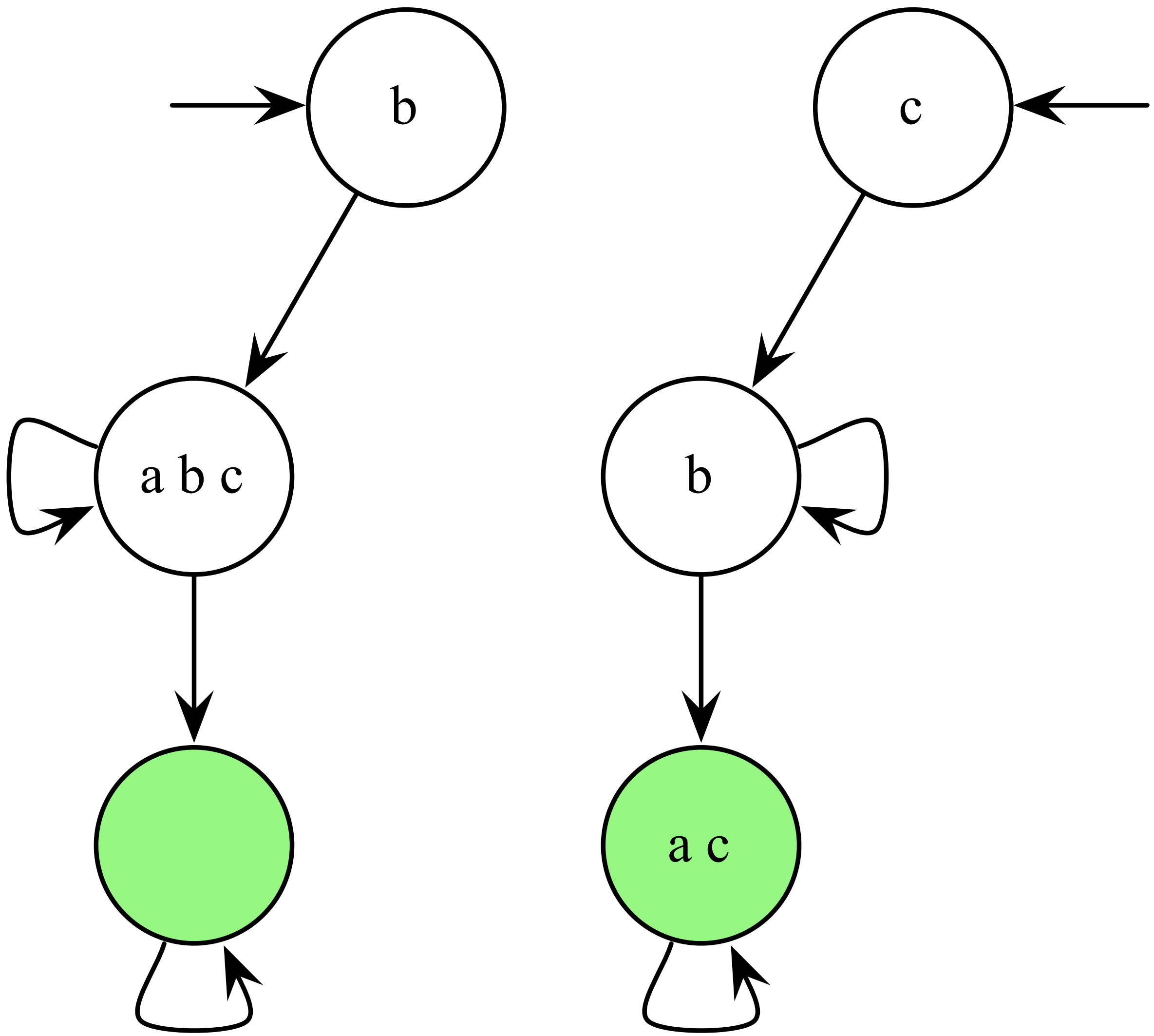
$$\widetilde{T}_{\psi} \models \varphi$$



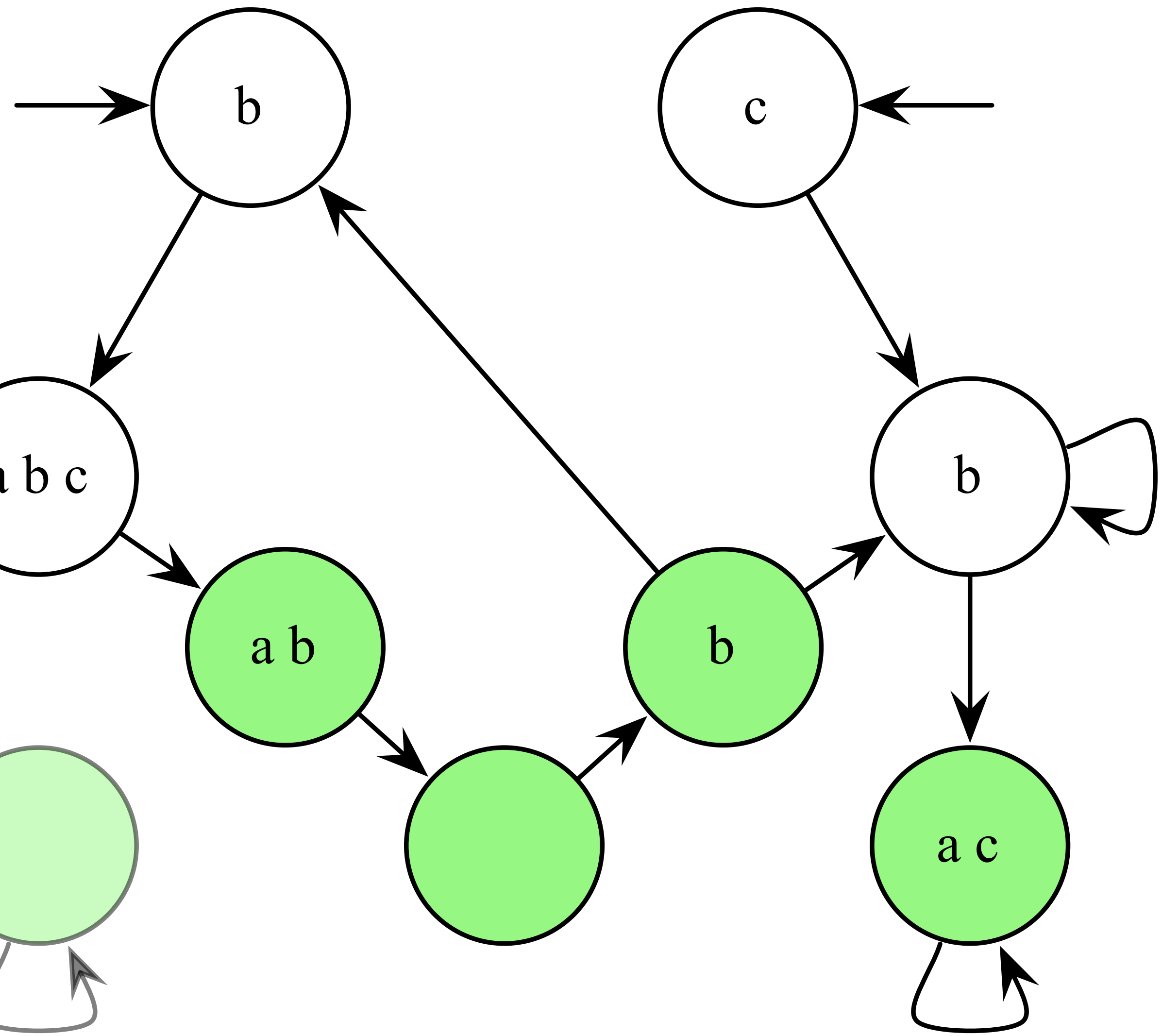
Result

- The aspect satisfies its specification

Really?



Really.



Aspect Verification

- Prove once-and-for-all that an aspect satisfies its specification
- Modular
- Generic
- Uses an LTL tableau as a “generic” model
- More on the way

Modular Generic Verification of LTL Properties for Aspects

Shmuel Katz
Max Goldman

{katz, mgoldman}@cs.technion.ac.il

FOAL ’06