

# The Power of Parameterization in Coinductive Proof

[Chung-Kil Hur](#)

Microsoft Research Cambridge

Georg Neis, Derek Dreyer, Viktor Vafeiadis

MPI-SWS

POPL 2013

23 Jan 2013

# Two Principles for Coinduction

- Tarski's Fixed Point Theorem
  - + Simple & Robust
  - Inconvenient to use
- Syntactically Guarded Coinduction
  - Complex & Fragile due to "Guardedness Checking"
  - + More convenient to use

# Our Contribution

- Parameterized Coinduction
  - + Simple & Robust
  - + Most convenient to use

Key Idea

Semantic Guardedness

- Parameterized Coinduction
  - + Simple & Robust
  - + Most convenient to use

# Talk Outline

## □ Previous Approaches



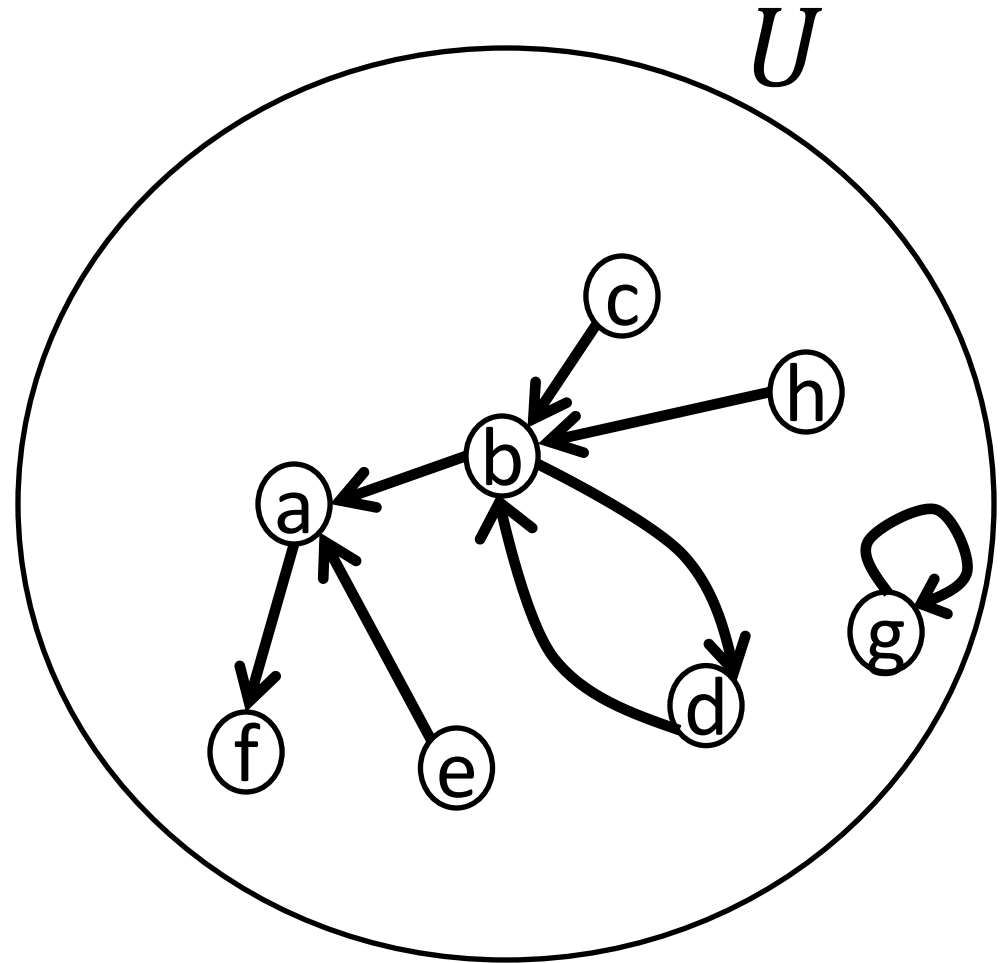
➤ Tarski's Fixed Point Theorem

➤ Syntactically Guarded Coinduction

## □ Our Approach

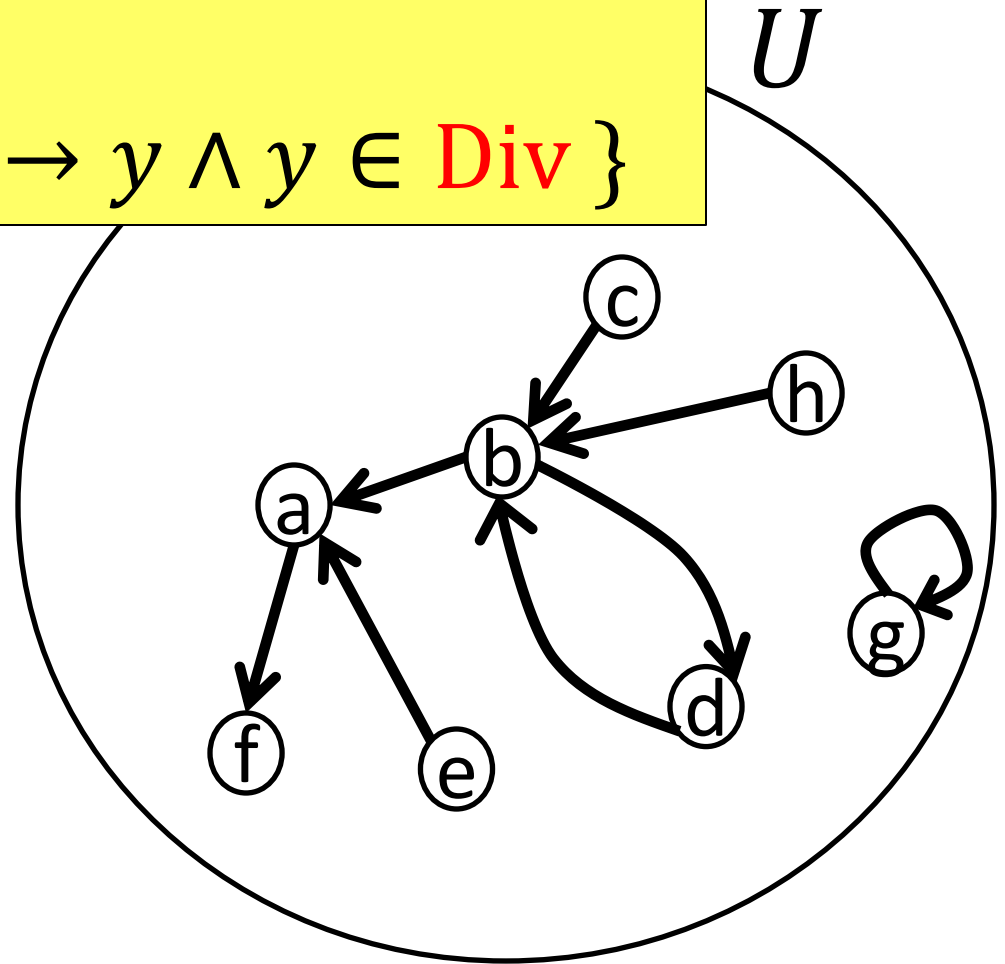
➤ Parameterized Coinduction

# Example of Recursively Defined Set: Divergent Nodes in STS



# Example of Recursively Defined Set: Divergent Nodes in STS

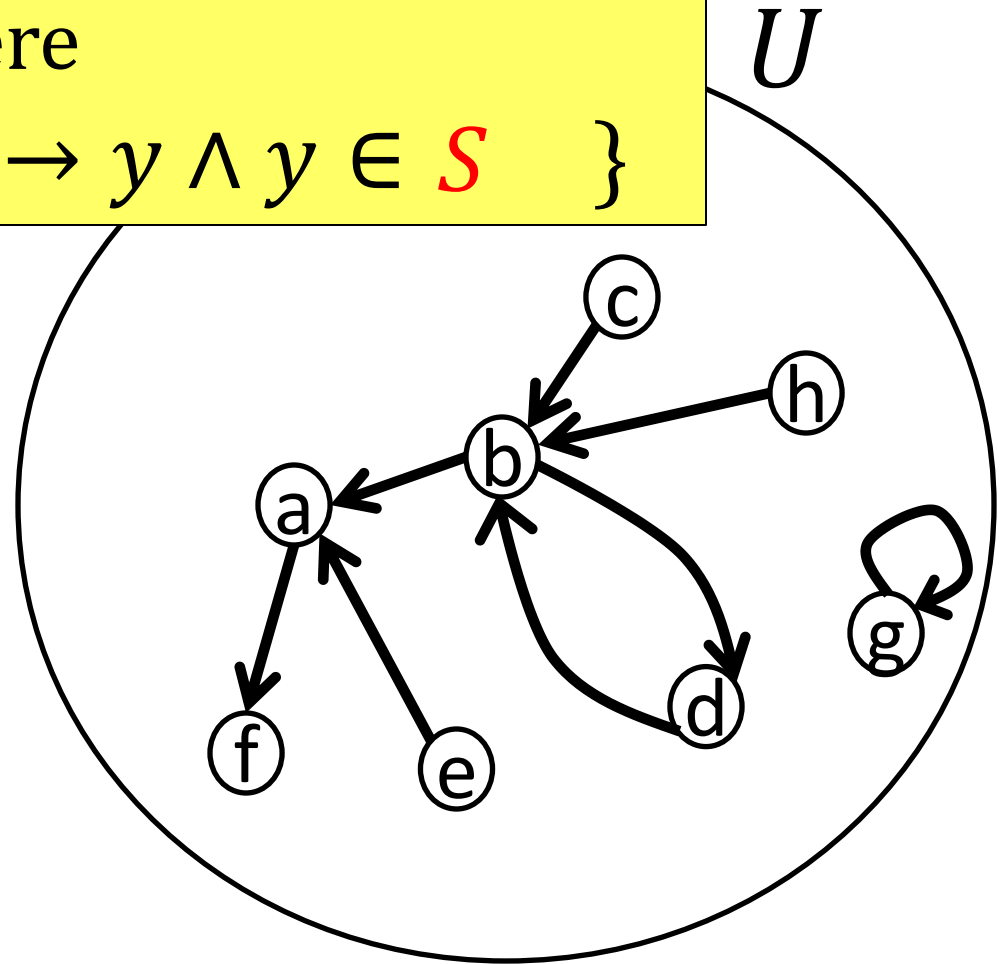
$$\text{Div} = \{ x \mid \exists y. x \rightarrow y \wedge y \in \text{Div} \}$$



# Example of Recursively Defined Set: Divergent Nodes in STS

**Div** =  $f(\mathbf{Div})$  where

$$f(S) = \{ x \mid \exists y. x \rightarrow y \wedge y \in S \}$$





# Example of Recursively Defined Set: Divergent Nodes in STS

$\text{Div} = f(\text{Div})$  where

$$f(S) = \{ x \mid \exists y. x \rightarrow y \wedge y \in S \}$$

$U$

(Tarski's Theorem) For  $f : \wp(U) \xrightarrow{\text{mon}} \wp(U)$ ,  
 $\nu f$  is the greatest solution of  $X = f(X)$

(f)

(e)

# Example of Recursively Defined Set: Divergent Nodes in STS

$\text{Div} = f(\text{Div})$  where

$$f(S) = \{ x \mid \exists y. x \rightarrow y \wedge y \in S \}$$

$U$

(Tarski's Theorem) For  $f : \wp(U) \xrightarrow{\text{mon}} \wp(U)$ ,  
 $\nu f$  is the greatest solution of  $X = f(X)$

$$\nu f = \bigcup \{ S \mid S \subseteq f(S) \}$$

$S$  is consistent

ⓔ

# Tarski's Principle

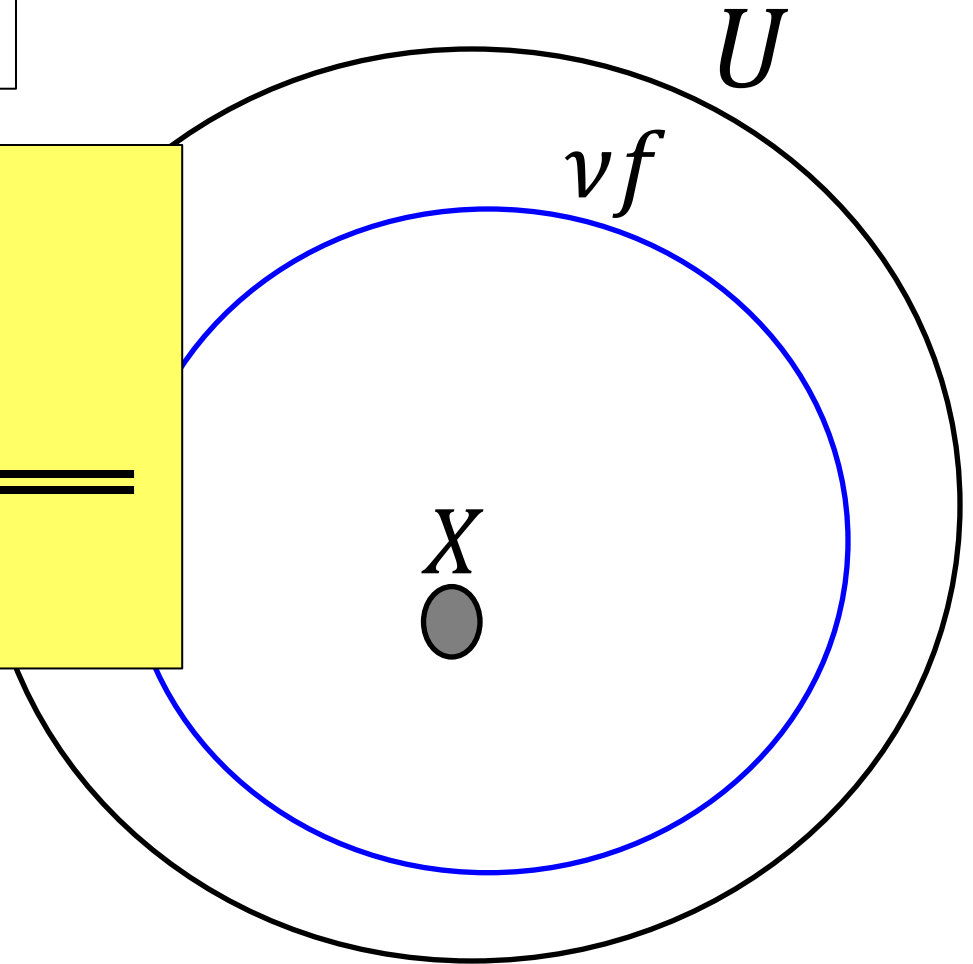
$$f : \wp(U) \xrightarrow{\text{mon}} \wp(U)$$

(Tarski's Principle)

---

---

$$X \subseteq \nu f$$



# Tarski's Principle

$$f : \wp(U) \xrightarrow{\text{mon}} \wp(U)$$

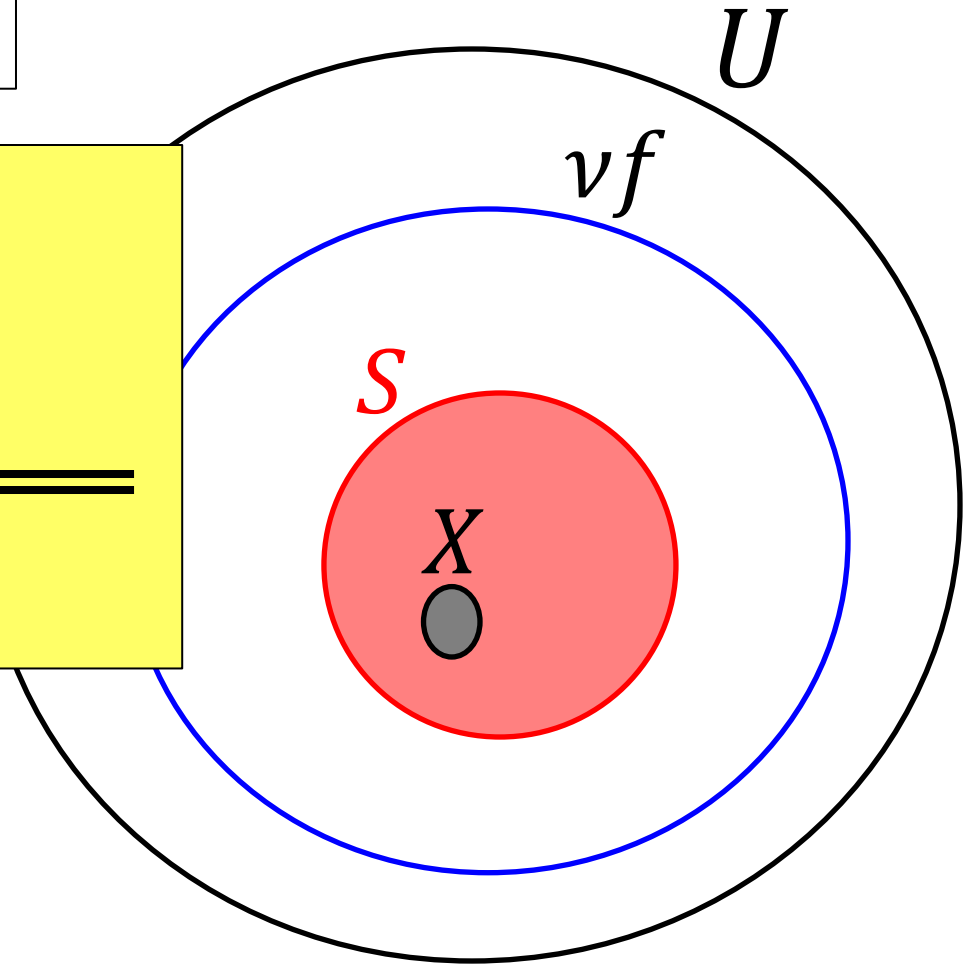
(Tarski's Principle)

$$\exists S. X \subseteq S$$

---

---

$$X \subseteq vf$$



# Tarski's Principle

$$f : \wp(U) \xrightarrow{\text{mon}} \wp(U)$$

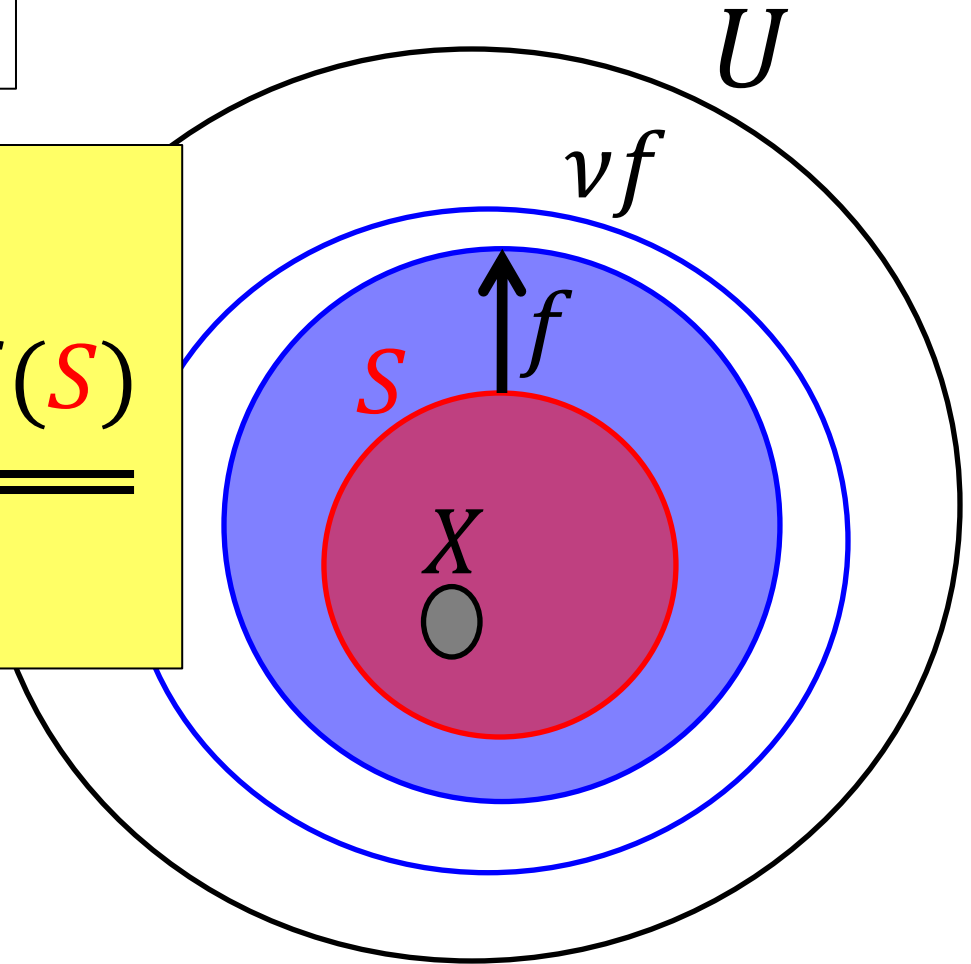
(Tarski's Principle)

$$\exists S. X \subseteq S \wedge S \subseteq f(S)$$

---

---

$$X \subseteq \nu f$$



# Tarski's Principle

Sound by Definition:  $\nu f = \bigcup \{ S \mid S \subseteq f(S) \}$

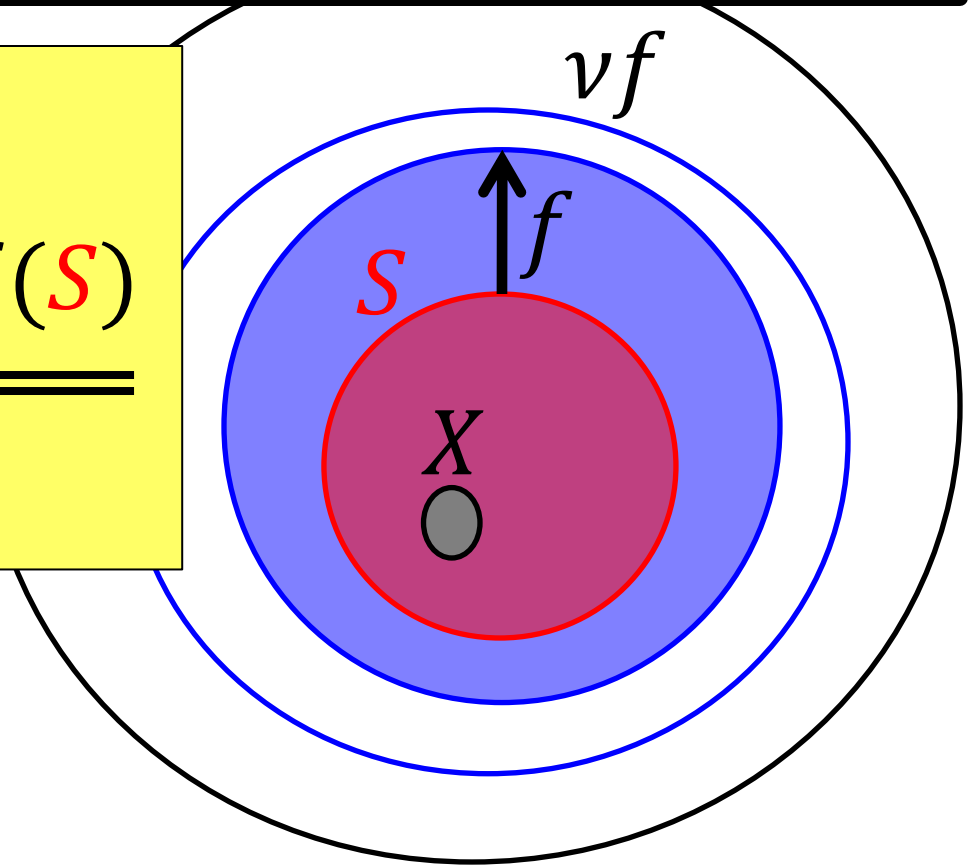
(Tarski's Principle)

$$\exists S. X \subseteq S \wedge S \subseteq f(S)$$

---

---

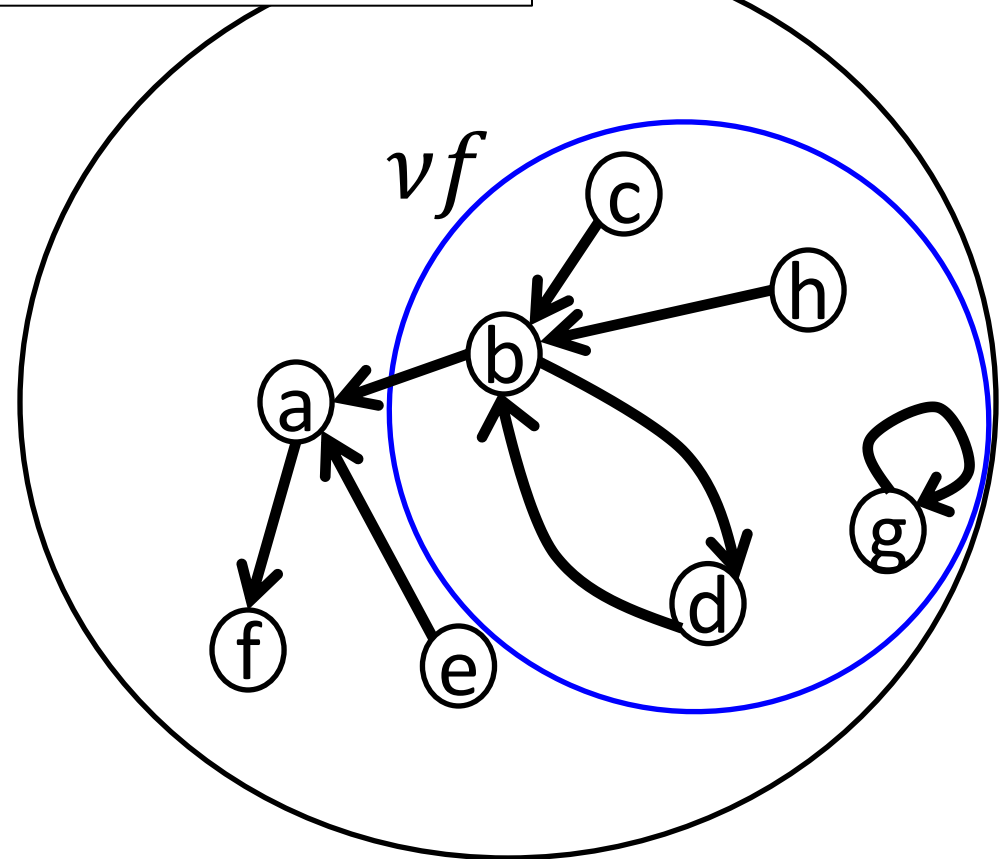
$$X \subseteq \nu f$$



# Tarski Principle: Example

$$f(S) = \{ x \mid \exists y. x \rightarrow y \wedge y \in S \}$$

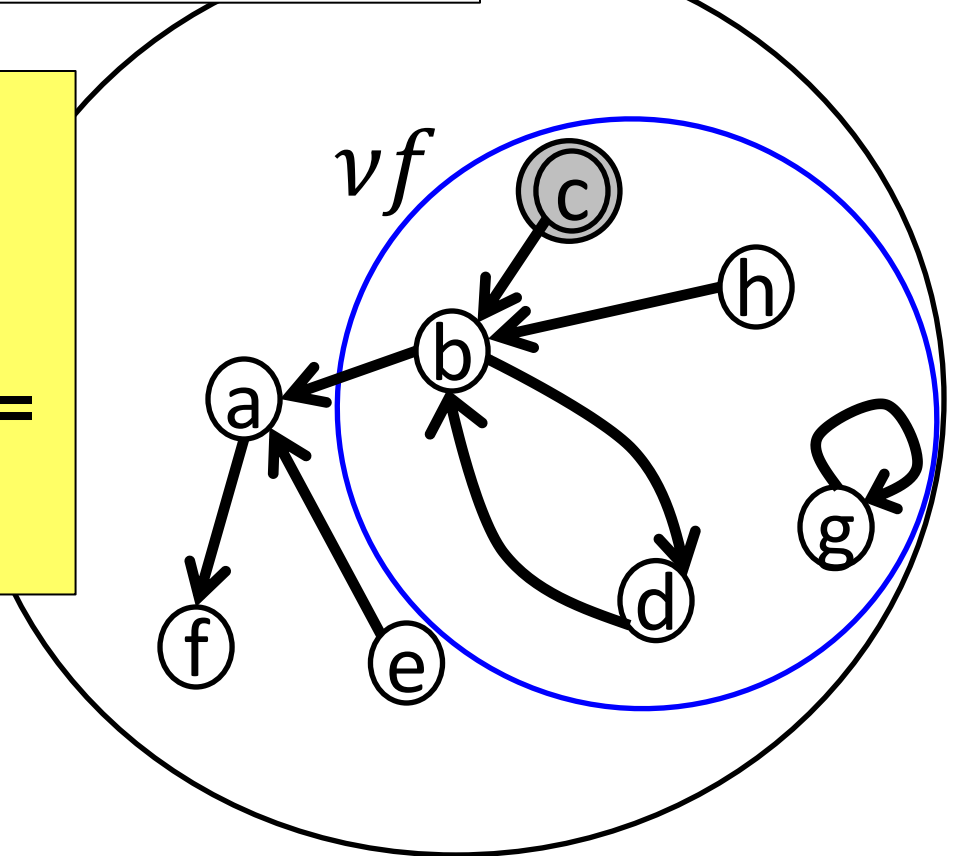
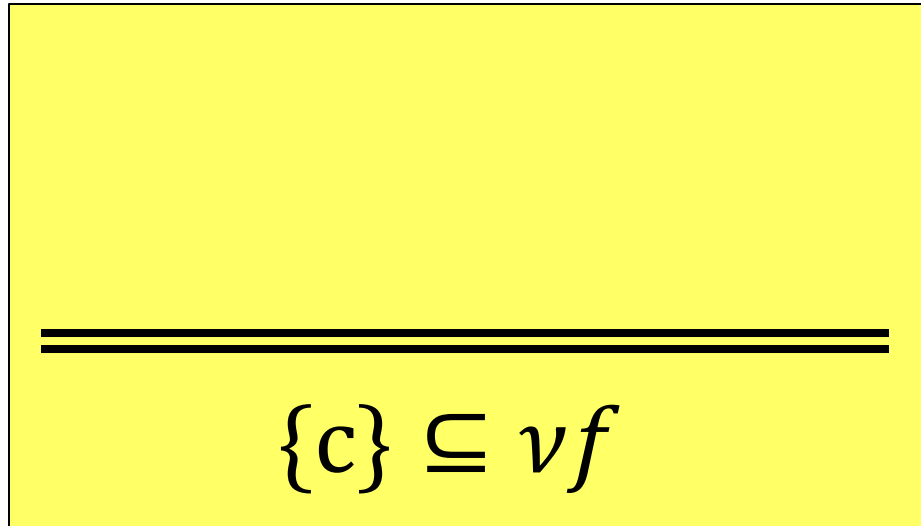
$U$



# Tarski Principle: Example

$$f(S) = \{x \mid \exists y. x \rightarrow y \wedge y \in S\}$$

$U$





# Tarski Principle: Example

$$f(S) = \{x \mid \exists y. x \rightarrow y \wedge y \in S\}$$

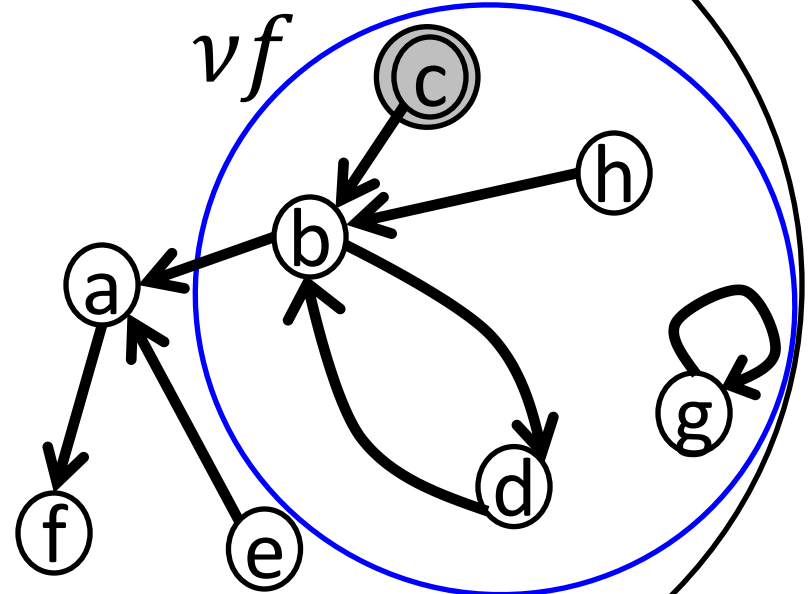
$U$

$$\{c\} \not\subseteq f(\{c\}) = \emptyset$$

---

---

$$\{c\} \subseteq vf$$



# Tarski Principle: Example

$$f(S) = \{x \mid \exists y. x \rightarrow y \wedge y \in S\}$$

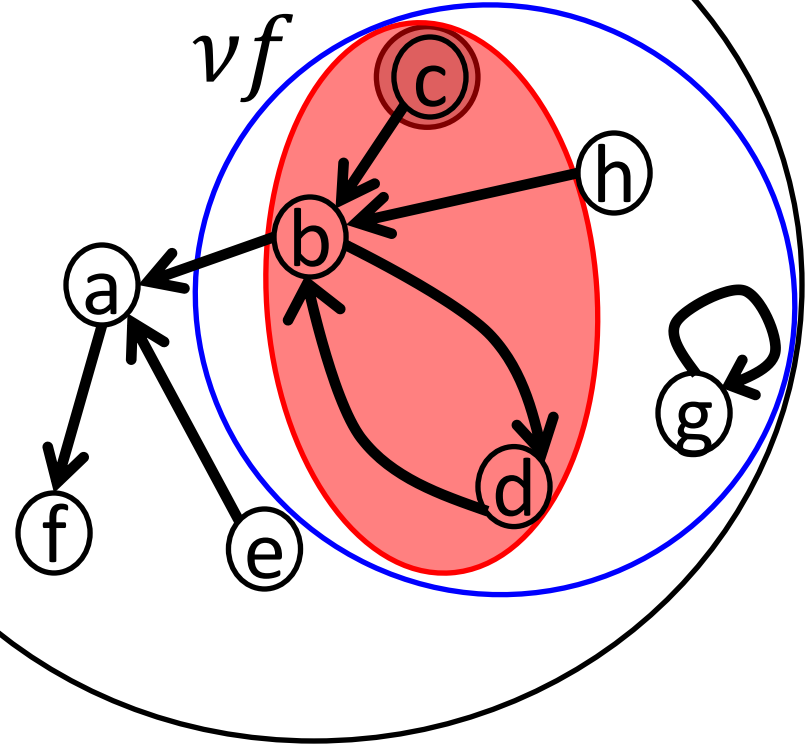
$U$

$$\{c\} \subseteq \{c, b, d\}$$

---

---

$$\{c\} \subseteq vf$$



# Tarski Principle: Example

$$f(S) = \{ x \mid \exists y. x \rightarrow y \wedge y \in S \}$$

$U$

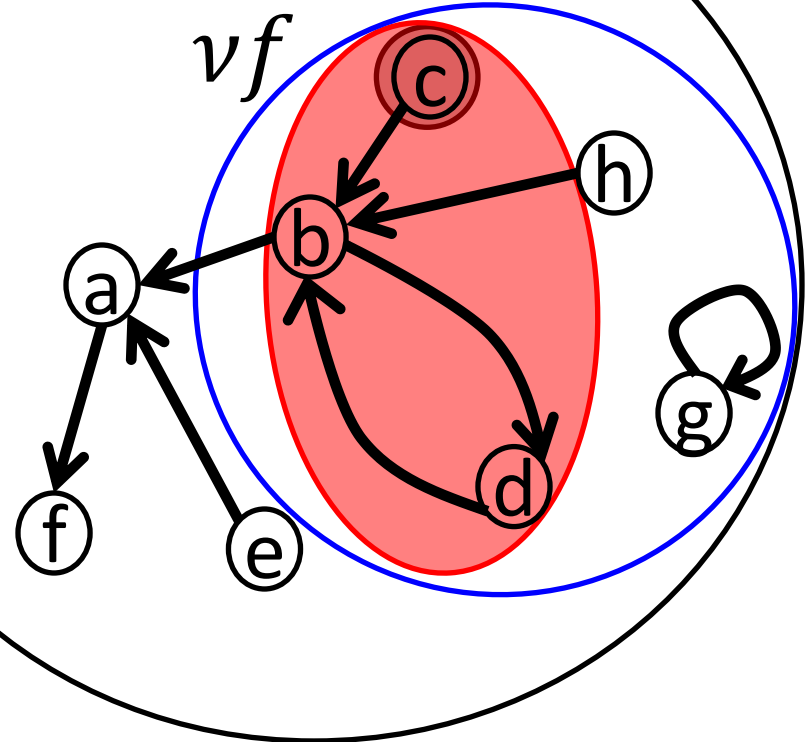
$$\{c\} \subseteq \{c, b, d\}$$

$$\{c, b, d\} \subseteq f(\{c, b, d\})$$

---

---

$$\{c\} \subseteq vf$$



# Tarski Principle: Example

$$f(S) = \{ x \mid \exists y. x \rightarrow y \wedge y \in S \}$$

$U$

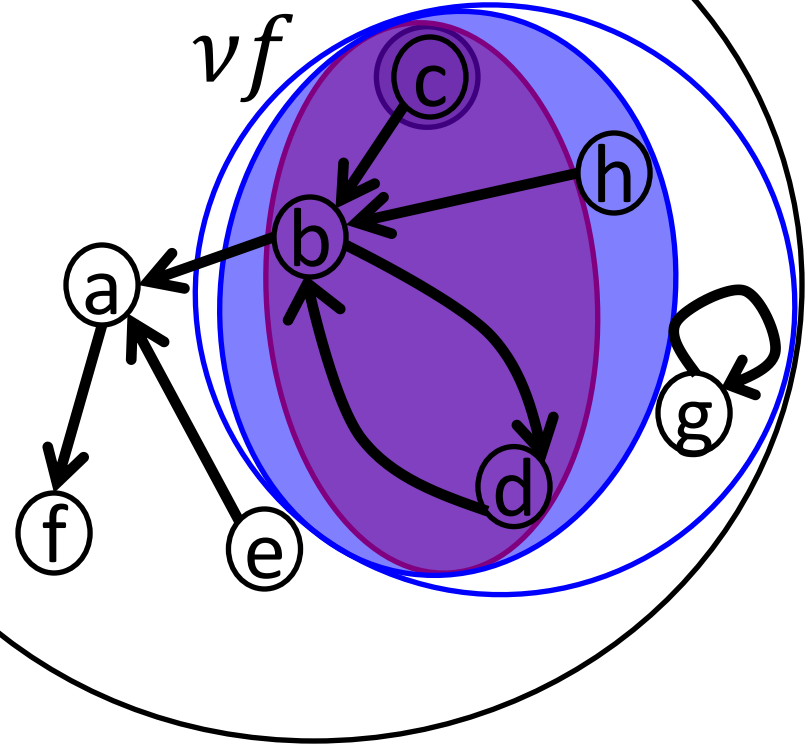
$$\{c\} \subseteq \{c, b, d\}$$

$$\{c, b, d\} \subseteq f(\{c, b, d\})$$

---

---

$$\{c\} \subseteq vf$$



# Tarski Principle: Example

$$f(S) = \{ x \mid \exists y. x \rightarrow y \wedge y \in S \}$$

$U$

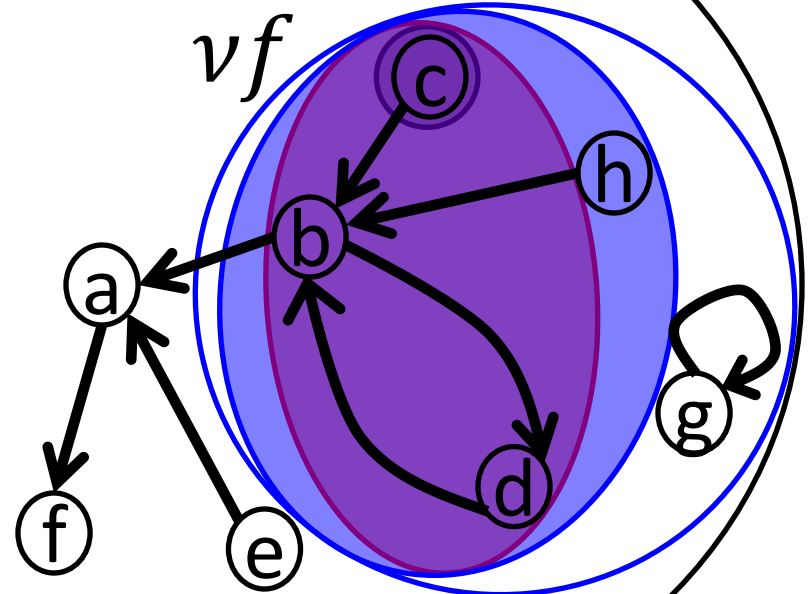
$$\{c\} \subseteq \{c, b, d\}$$

$$\{c, b, d\} \checkmark \equiv f(\{c, b, d\})$$

---

---

$$\{c\} \subseteq vf$$



# Tarski's Principle

+ Simple & Easy to Understand

- Have to Find a Consistent Set Up Front

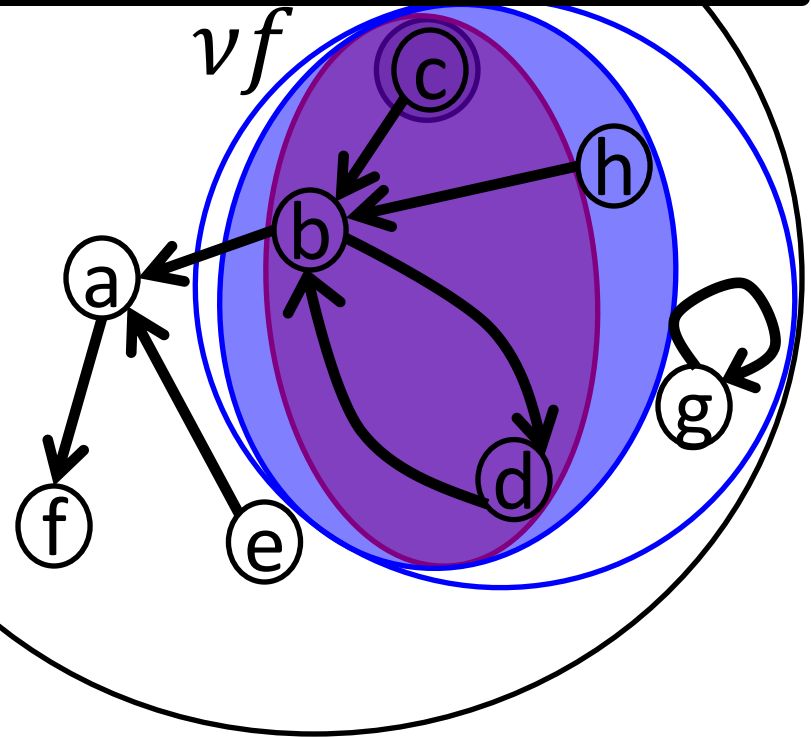
$$\{c\} \subseteq \{c, b, d\}$$

$$\{c, b, d\} \checkmark \equiv f(\{c, b, d\})$$

---

---

$$\{c\} \subseteq vf$$



# Talk Outline

- Previous Approaches

- Tarski's Fixed Point Theorem



- Syntactically Guarded Coinduction

- Our Approach

- Parameterized Coinduction

# Syntactically Guarded Coinduction

$$f : \wp(U) \xrightarrow{\text{mon}} \wp(U)$$

(Guarded Coinduction)

---

$$X \subseteq \nu f$$



# Syntactically Guarded Coinduction

$$f : \wp(U) \xrightarrow{\text{mon}} \wp(U)$$

(Guarded Coinduction)

$$X \subseteq \nu f \implies X \subseteq \nu f$$

---

$$X \subseteq \nu f$$

# Syntactically Guarded Coinduction

$$f : \wp(U) \xrightarrow{\text{mon}} \wp(U)$$

(Guarded Coinduction)

$$\frac{\text{pf: } X \subseteq \nu f \implies X \subseteq \nu f}{X \subseteq \nu f} \quad (\text{pf is guarded})$$

# Syntactically Guarded Coinduction

$f$

- Guardedness rules out trivial proofs.

(Gu

pf:  $X \subseteq \nu f \Rightarrow X \subseteq \nu f$

---

$X \subseteq \nu f$

(pf is guarded)

# Syntactically Guarded Coinduction

$f$

- Guardedness **rules out trivial proofs.**
- Its definition is **syntactic** and **complex.**

(Gu

pf:  $X \subseteq \nu f \Rightarrow X \subseteq \nu f$

---

$X \subseteq \nu f$

(pf is **guarded**)

# Syntactically Guarded Coinduction

$f$

- Guardedness **rules out trivial proofs.**
- Its definition is **syntactic** and **complex.**
- The assumption can be used **only after making progress.**

(Gu

pf:  $X \subseteq \nu f \Rightarrow X \subseteq \nu f$

---

$X \subseteq \nu f$

(pf is **guarded**)

Syn

duction

What **cofix** does in Coq

ofs.

$f$

- Its definition is **syntactic** and **complex**.
- The assumption can be used **only after making progress**.

(Gu

pf:  $X \subseteq \nu f \Rightarrow X \subseteq \nu f$

(pf is **guarded**)

$X \subseteq \nu f$

Syn

duction

Permits **Incremental** Reasoning

ofs.

$f$

- Its definition is **syntactic** and **complex**.
- The assumption can be used **only after making progress**.

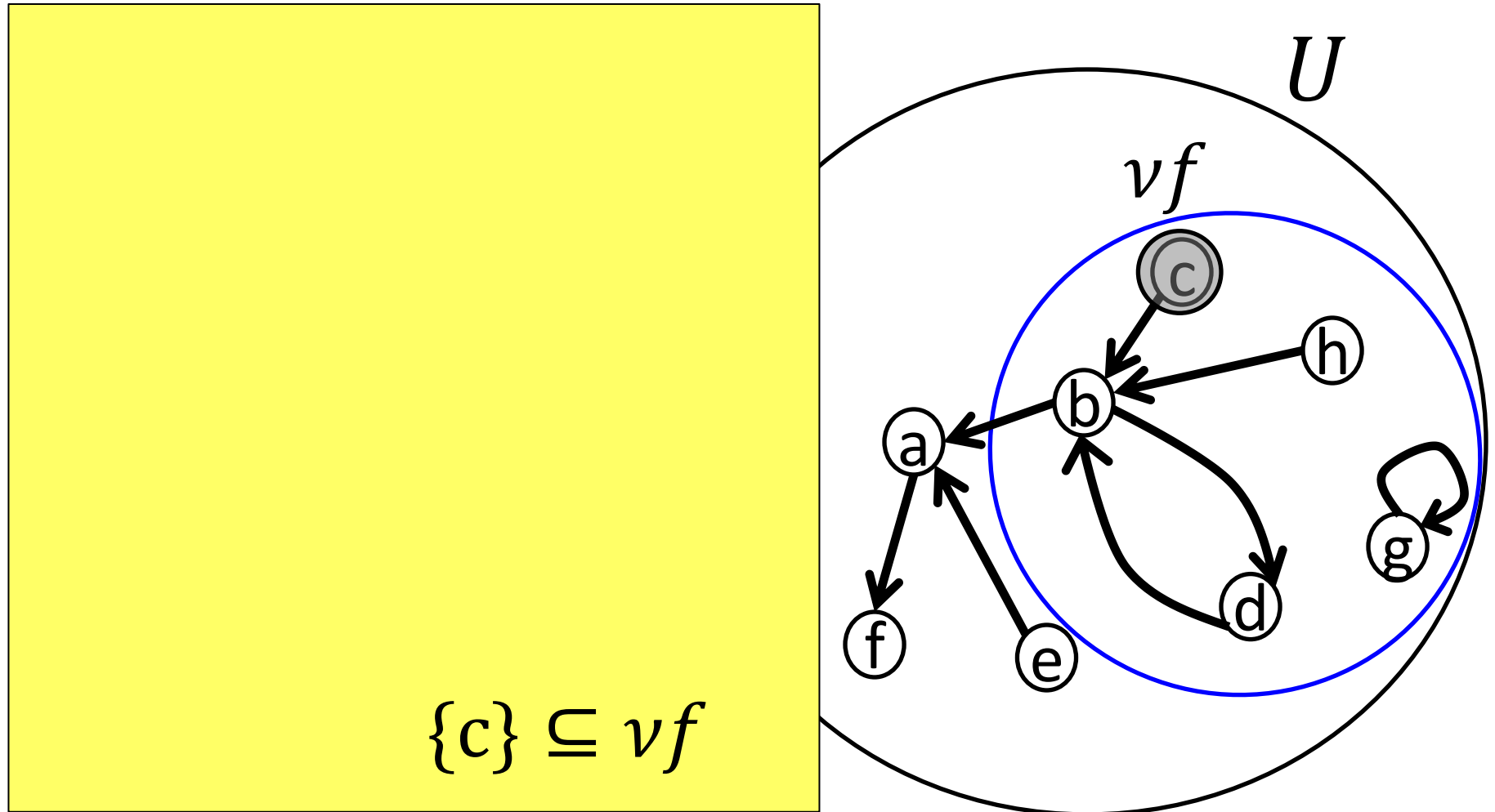
(Gu

pf:  $X \subseteq \nu f \Rightarrow X \subseteq \nu f$

(pf is **guarded**)

$X \subseteq \nu f$

# Example: Guarded Coinduction



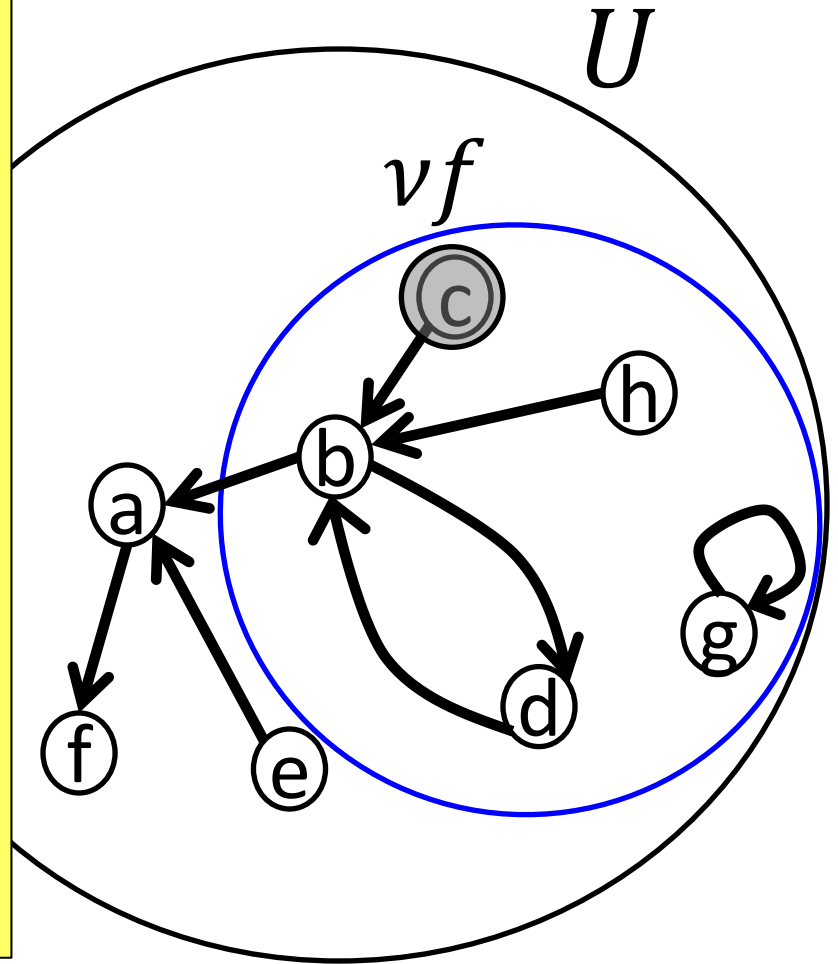


# Example: Guarded Coinduction

$$\{c\} \subseteq f(vf)$$

---

$$\{c\} \subseteq vf$$

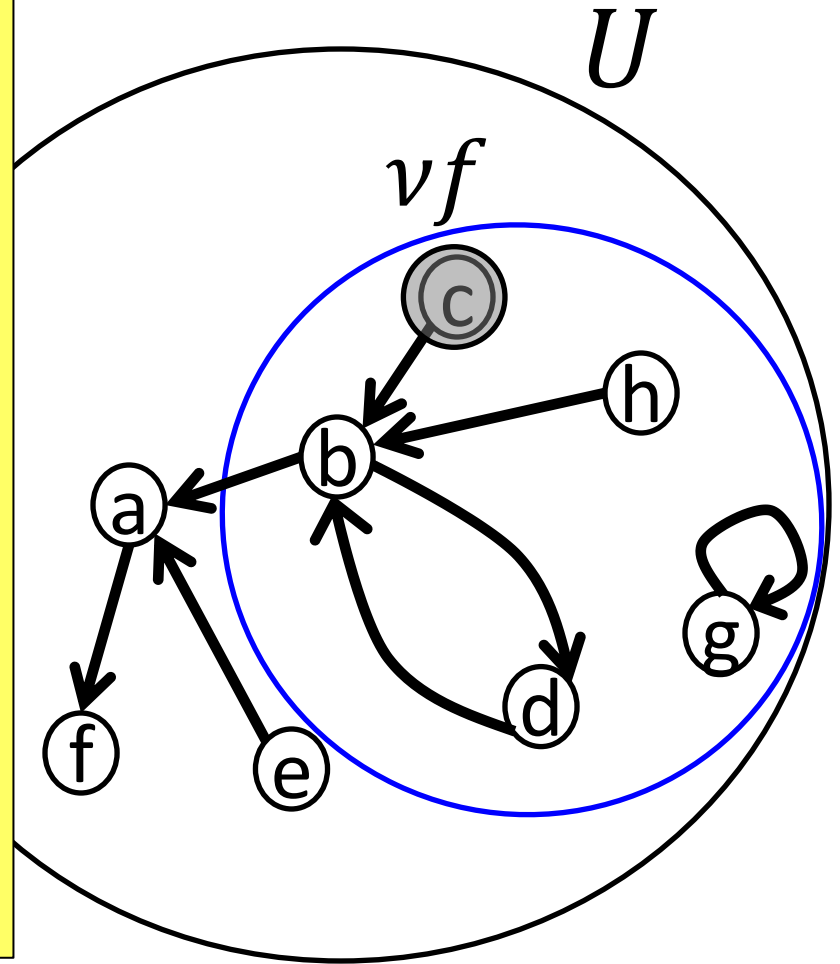


# Example: Guarded Coinduction

$$\exists y. c \rightarrow y \wedge y \in vf$$

---

$$\{c\} \subseteq vf$$

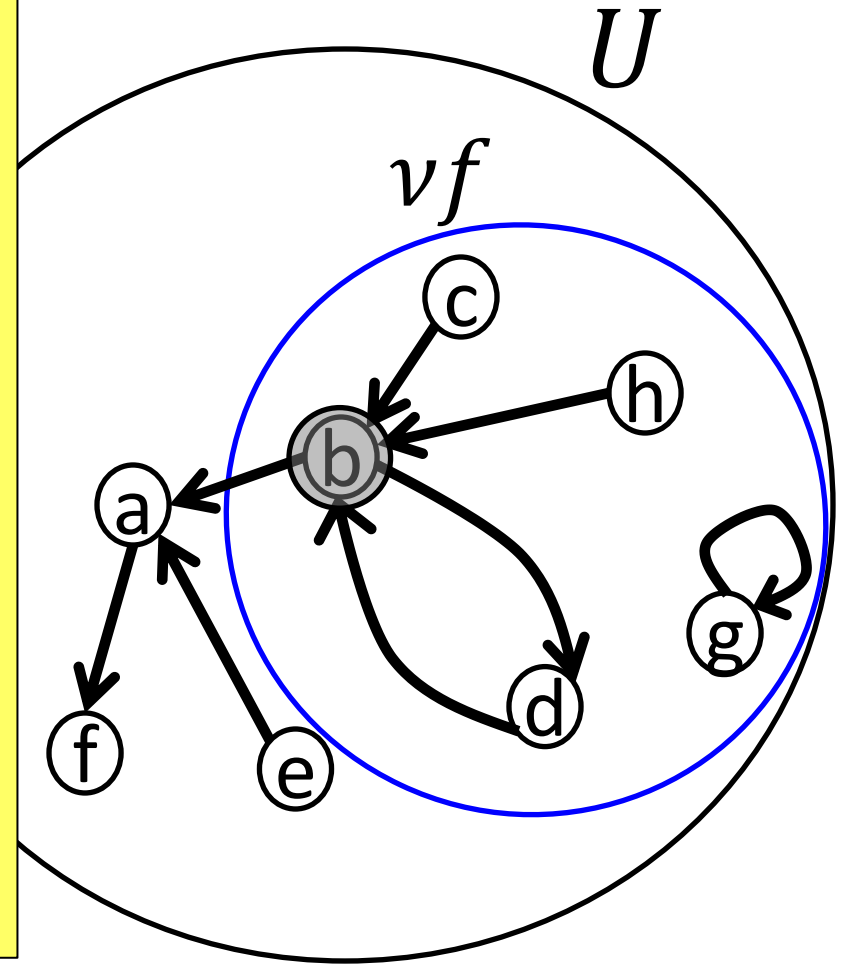


# Example: Guarded Coinduction

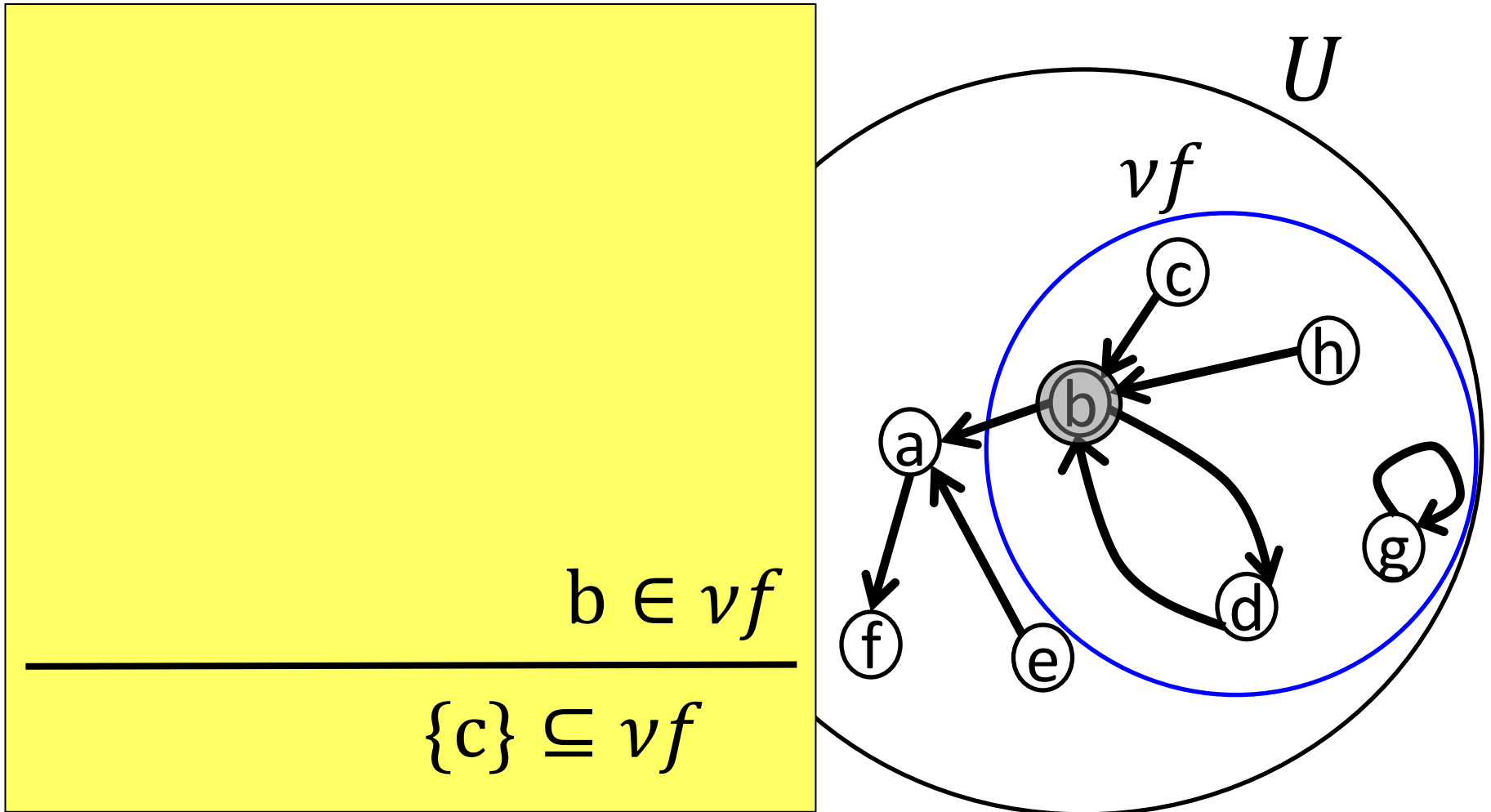
$$\exists y. c \rightarrow y \wedge y \in vf$$

---

$$\{c\} \subseteq vf$$



# Example: Guarded Coinduction



# Example: Guarded Coinduction

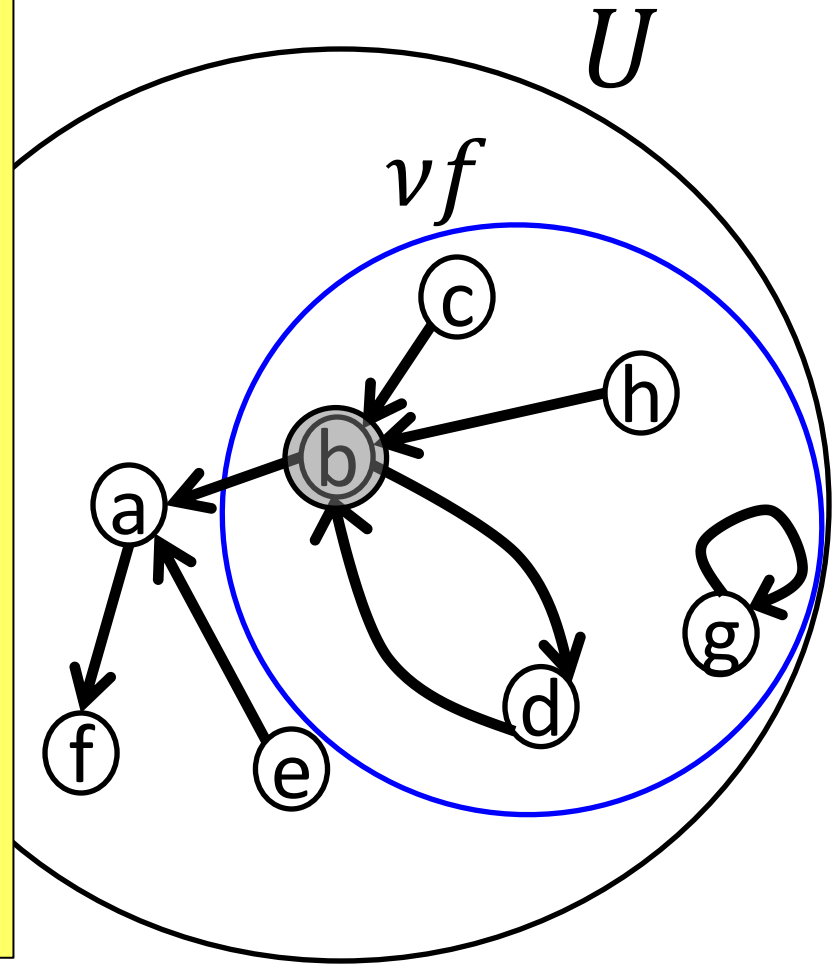
$$\{b\} \subseteq \nu f$$

---

$$b \in \nu f$$

---

$$\{c\} \subseteq \nu f$$



# Example: Guarded

By  
Guarded Coinduction

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

---

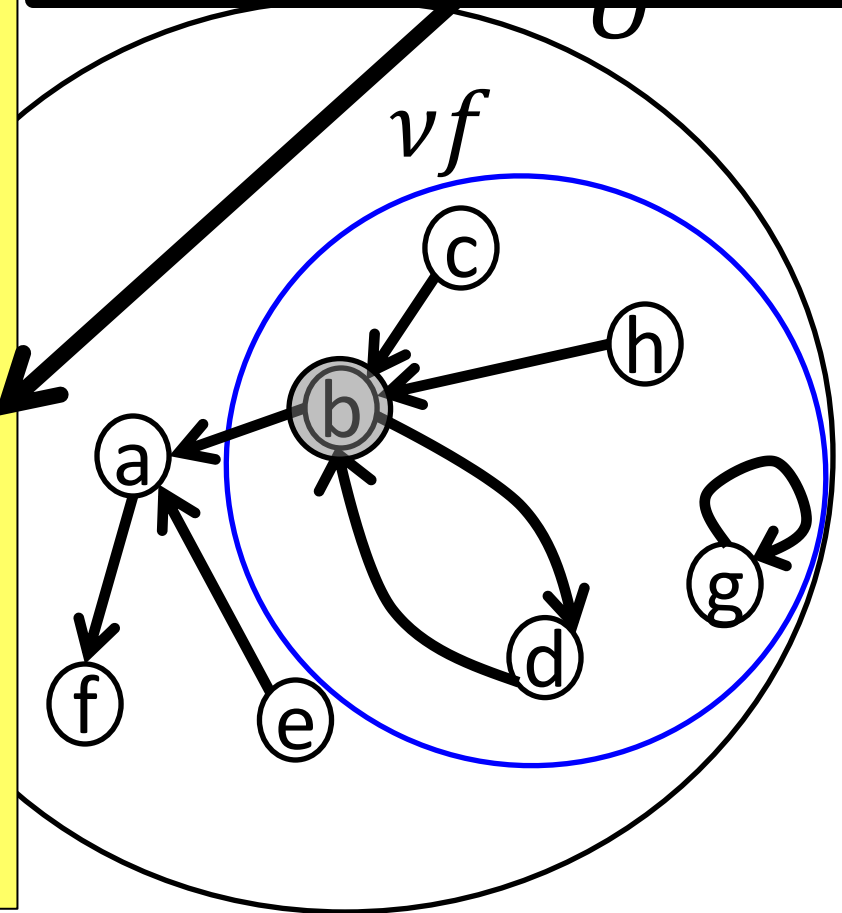
$$\{b\} \subseteq \nu f$$

---

$$b \in \nu f$$

---

$$\{c\} \subseteq \nu f$$



# Example: Guarded

By  
Guarded Coinduction

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

---

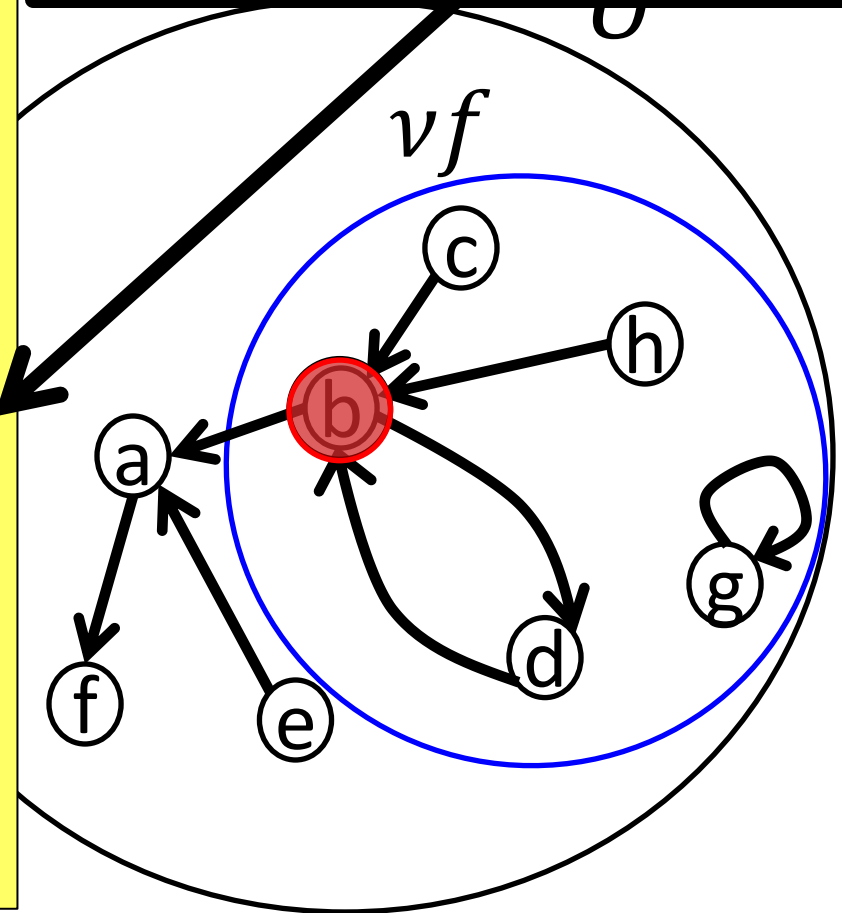
$$\{b\} \subseteq \nu f$$

---

$$b \in \nu f$$

---

$$\{c\} \subseteq \nu f$$



Trivial Proof is  
NOT Guarded

de

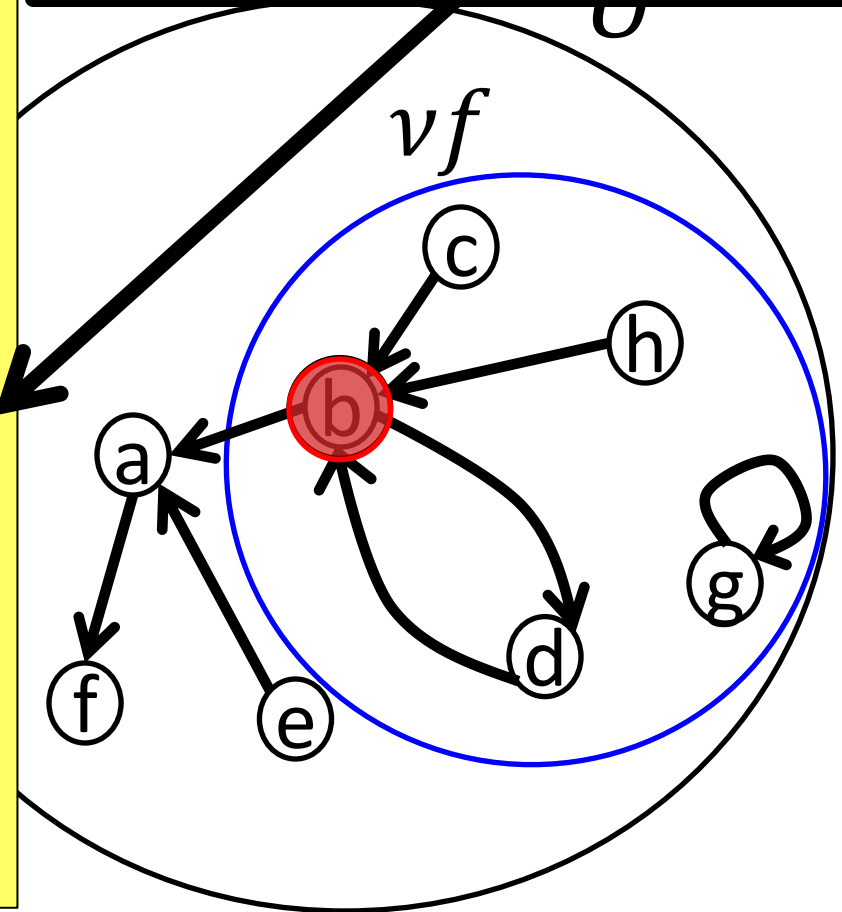
By  
Guarded Coinduction

$$\{b\} \subseteq vf \Rightarrow \{b\} \subseteq vf$$

$$\{b\} \subseteq vf$$

$$b \in vf$$

$$\{c\} \subseteq vf$$





# Example: Guarded

By  
Guarded Coinduction

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

---

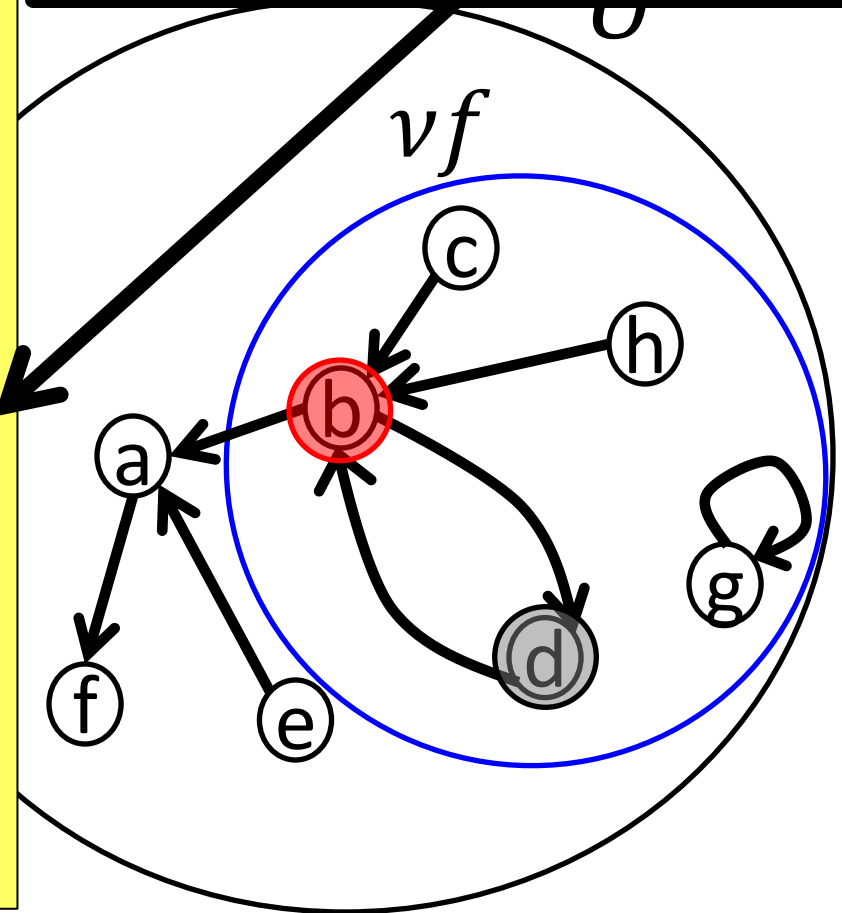
$$\{b\} \subseteq \nu f$$

---

$$b \in \nu f$$

---

$$\{c\} \subseteq \nu f$$



# Example: Guarded

By  
Guarded Coinduction

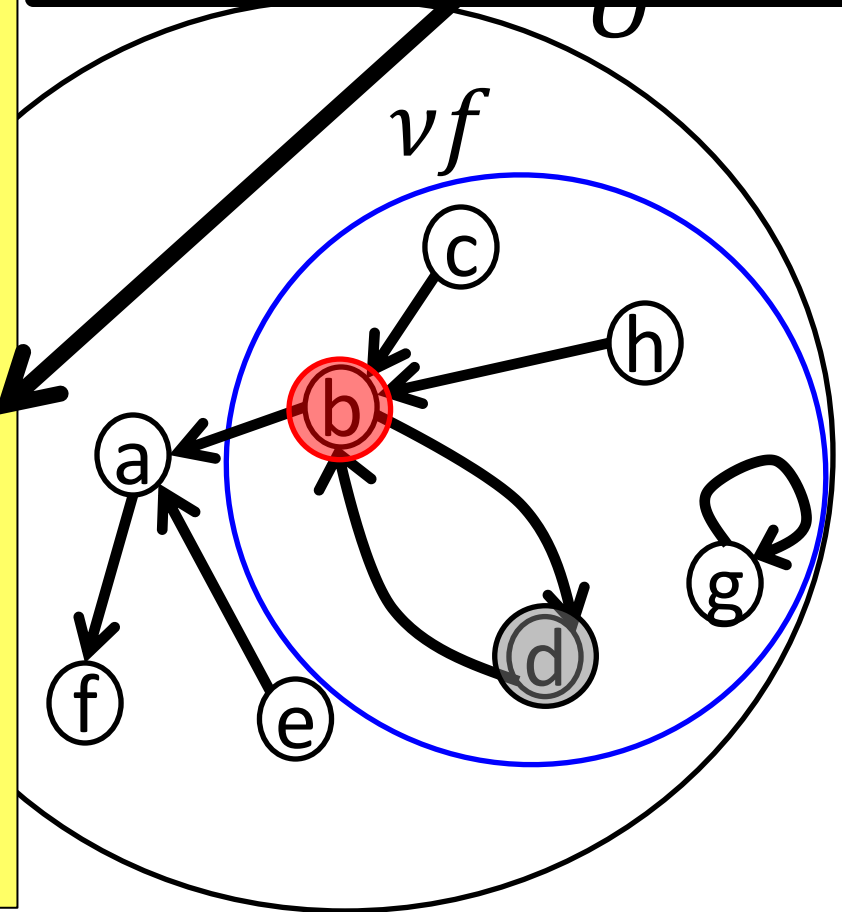
$$\{b\} \subseteq vf \Rightarrow \{d\} \subseteq vf$$

$$\{b\} \subseteq vf \Rightarrow \{b\} \subseteq vf$$

$$\{b\} \subseteq vf$$

$$b \in vf$$

$$\{c\} \subseteq vf$$



# Example: Guarded

By  
Guarded Coinduction

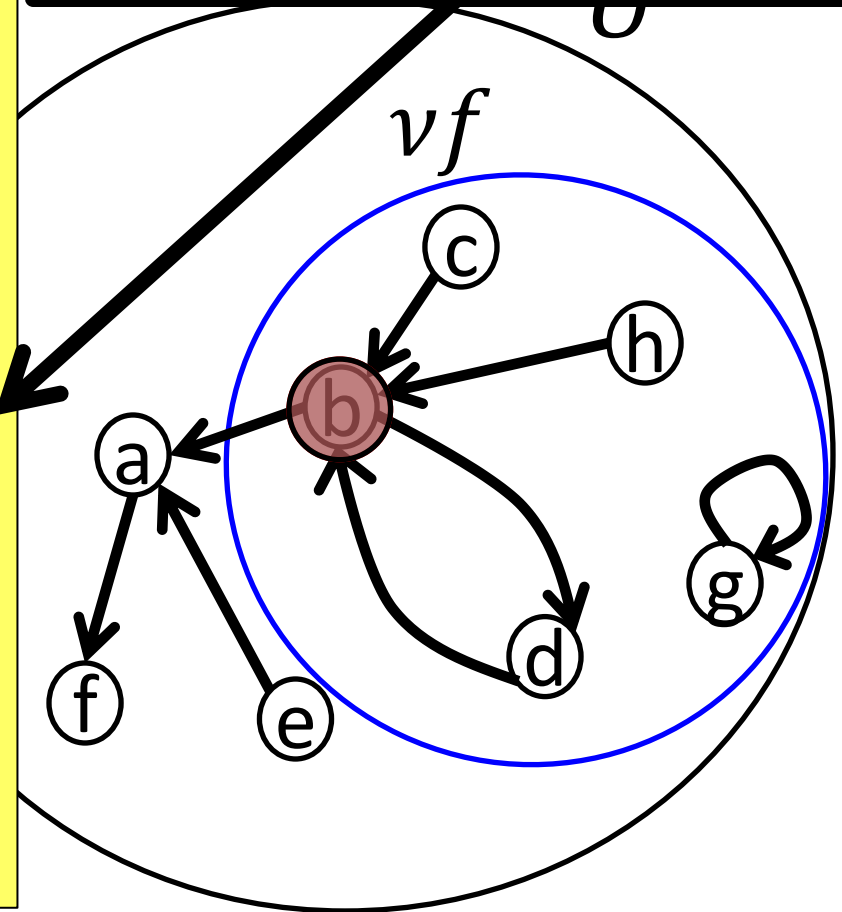
$$\{b\} \subseteq vf \Rightarrow \{d\} \subseteq vf$$

$$\{b\} \subseteq vf \Rightarrow \{b\} \subseteq vf$$

$$\{b\} \subseteq vf$$

$$b \in vf$$

$$\{c\} \subseteq vf$$



# Example: Guarded

By  
Guarded Coinduction

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

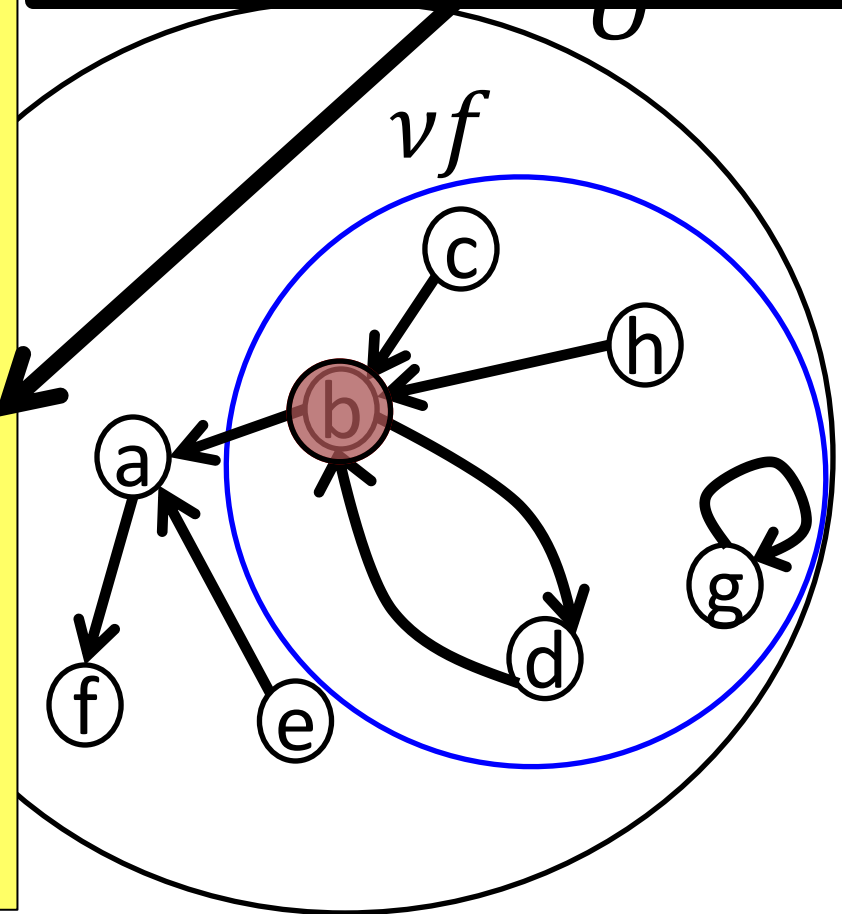
$$\{b\} \subseteq \nu f \Rightarrow \{d\} \subseteq \nu f$$

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

$$\{b\} \subseteq \nu f$$

$$b \in \nu f$$

$$\{c\} \subseteq \nu f$$



This Proof is  
Guarded

By  
Guarded Coinduction

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

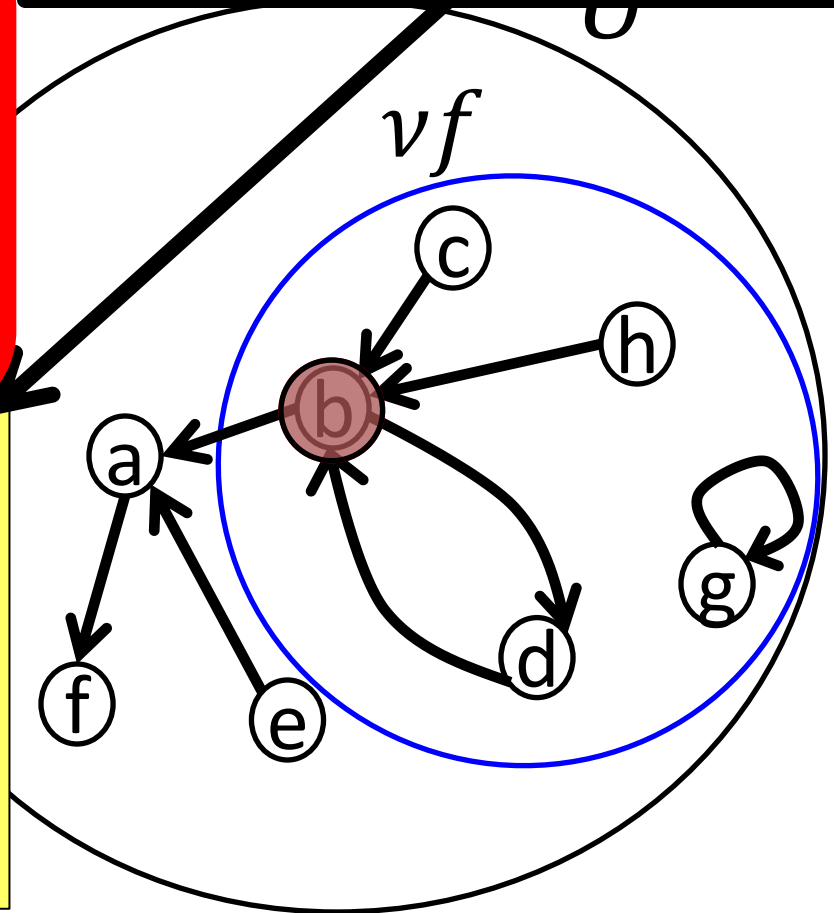
$$\{b\} \subseteq \nu f \Rightarrow \{d\} \subseteq \nu f$$

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

$$\{b\} \subseteq \nu f$$

$$b \in \nu f$$

$$\{c\} \subseteq \nu f$$



This Proof is  
Guarded

By  
Guarded Coinduction

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

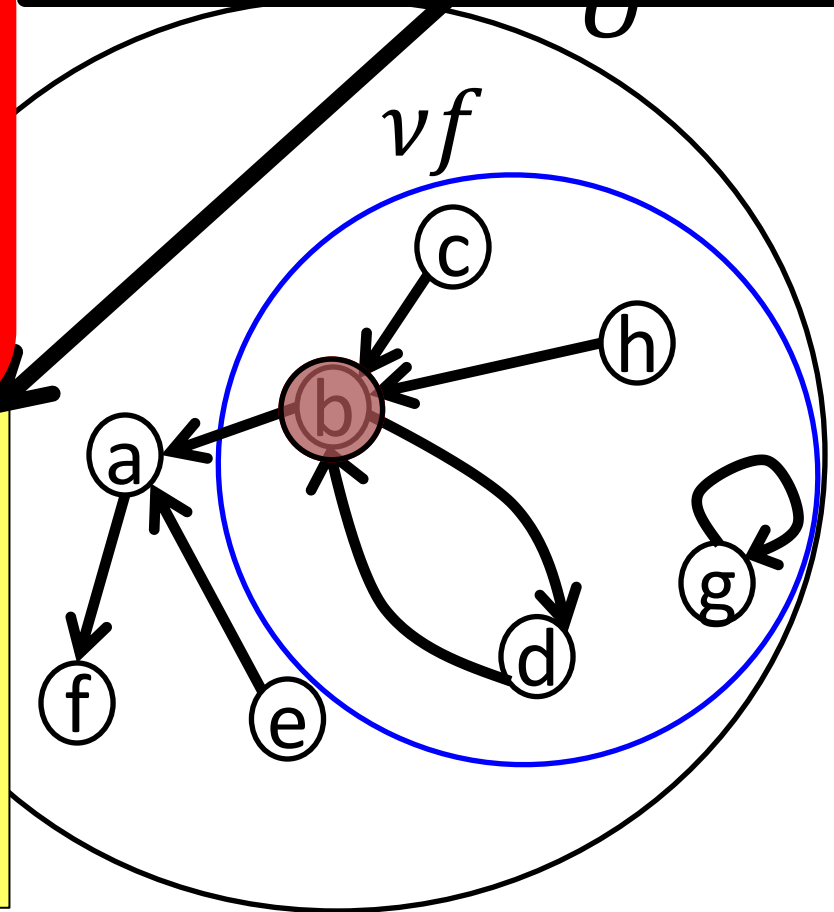
$$\{b\} \subseteq \nu f \Rightarrow \{d\} \subseteq \nu f$$

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

$$\{b\} \subseteq \nu f$$

$$b \in \nu f$$

$$\{c\} \subseteq \nu f$$



This Proof is  
**Guarded**

By  
Guarded Coinduction

$$\{b\} \subseteq \nu f \Rightarrow \{b\} \subseteq \nu f$$

Prove **Incrementally**  
with **NO Consistent Set** Up Front

$$\{c\} \subseteq \nu f$$

# Example: Simulation between Programs

restart  $h\ n :=$  if  $n > \underline{0}$  then (output  $n$ ; restart  $h\ (n - \underline{1})$ ) else  $h\ \underline{0}$

$f\ n :=$  output  $(\underline{2} * n)$ ;  
let  $v =$  input() in  
if  $v = \underline{0}$  then restart  $f\ (\underline{2} * n)$  else  $f\ (v + n)$

$g\ n :=$  let  $v =$  (output  $n$ ; input())  $* \underline{2}$  in  
(if  $v \neq \underline{0}$  then  $g$  else restart  $g$ )  $(v + n)$



# Example: Simulation between Programs

**restart**  $h\ n :=$  if  $n > \underline{0}$  then (output  $n$ ; **restart**  $h\ (n - \underline{1})$ ) else  $h\ \underline{0}$

**f**  $n :=$  output ( $\underline{2} * n$ );  
let  $v =$  input() in  
if  $v = \underline{0}$  then **restart** **f** ( $\underline{2} * n$ ) else **f** ( $v + n$ )

**g**  $n :=$  let  $v =$  (output  $n$ ; input()) \*  $\underline{2}$  in  
(if  $v \neq \underline{0}$  then **g** else **restart** **g**) ( $v + n$ )

# Proof using Tarski's Principle

```

inductive fgsim : exp -> exp -> Prop :=
  _fgsim1: forall (n m: nat) (EQ: n = 2 * m), fgsim
    (f @ n)
    (g @ m)
  _fgsim2: forall (m: nat), fgsim
    (Let "v" := (Eoutput (m + (m + 0)));; Einput) ~* 2
    in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))) |
    (Eoutput (2 ~* m));;
    (Let "v" := Einput
     in If "v" ~<= 0 then (restart @ g) @ (2 ~* m) else g @ ("v" ~+ m)))
  _fgsim3: forall (m: nat), fgsim
    (Let "v" := (<>);; Einput) ~* 2
    in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))) |
    (<>);;
    (Let "v" := Einput
     in If "v" ~<= 0 then (restart @ g) @ (2 ~* m) else g @ ("v" ~+ m)))
  _fgsim4: forall (m: nat), fgsim
    (Let "v" := Einput ~* 2
     in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))))
    (Let "v" := Einput
     in If "v" ~<= 0 then (restart @ g) @ (2 ~* m) else g @ ("v" ~+ m))
  _fgsim5: forall (m m0: nat), fgsim
    (Let "v" := m0 ~* 2
     in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))))
    (Let "v" := m0
     in If "v" ~<= 0 then (restart @ g) @ (2 ~* m) else g @ ("v" ~+ m))
  _fgsim6: forall (m m0: nat), fgsim
    (Let "v" := m0 * 1 + m0
     in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))) |
     (If m0 ~<= 0 then (restart @ g) @ (2 ~* m) else g @ (m0 ~+ m)))
  _fgsim7: forall (m m0: nat), fgsim
    ((If 1 ~<= (m0 * 1 + m0) then f else restart @ f) @
     (m0 * 1 + m0) ~+ (m + (m + 0))) |
    (If ble_nat m0 0 then (restart @ g) @ (2 ~* m) else g @ (m0 ~+ m))
  _fgsim8: forall (m m0: nat), fgsim
    ((If 1 ~<= (m0 * 1 + m0) then f else restart @ f) @
     (m0 * 1 + m0 + (m + (m + 0)))) |
    (If ble_nat m0 0 then (restart @ g) @ (2 ~* m) else g @ (m0 ~+ m))
  _fgsim9: forall (m m0: nat), fgsim
    ((If ble_nat 1 (m0 * 1 + m0) then f else restart @ f) @
     (m0 * 1 + m0 + (m + (m + 0)))) |
    (If ble_nat m0 0 then (restart @ g) @ (2 ~* m) else g @ (m0 ~+ m))
  _fgsim10: forall (m: nat), fgsim
    ((restart @ f) @ m) ((restart @ g) @ m)
  _fgsim11: forall (m: nat), fgsim
    (Efix " " "n"
     (If "n" ~<= 0 then f @ 0
      else Eoutput "n";; (restart @ f) @ ("n" ~- 1)) @ m)
    (Efix " " "n"
     (If "n" ~<= 0 then g @ 0
      else Eoutput "n";; (restart @ g) @ ("n" ~- 1)) @ m)
  _fgsim12: forall (m: nat), fgsim
    (If m ~<= 0 then f @ 0 else Eoutput m;; (restart @ f) @ (m ~- 1))
    (If m ~<= 0 then g @ 0 else Eoutput m;; (restart @ g) @ (m ~- 1))
  _fgsim13: forall (m: nat), fgsim
    (If ble_nat m 0 then f @ 0 else Eoutput m;; (restart @ f) @ (m ~- 1))
    (If ble_nat m 0 then g @ 0 else Eoutput m;; (restart @ g) @ (m ~- 1))
  _fgsim14: forall (m: nat), fgsim
    (Eoutput (S m);; (restart @ f) @ (S m ~- 1))
    (Eoutput (S m);; (restart @ g) @ (S m ~- 1))
  _fgsim15: forall (m: nat), fgsim
    (<>);; (restart @ f) @ (S m ~- 1)
    (<>);; (restart @ g) @ (S m ~- 1)
  _fgsim16: forall (m: nat), fgsim
    ((restart @ f) @ (S m ~- 1))
    ((restart @ g) @ (S m ~- 1))
  :
  Lemma rsp_simulated_tarski:
    forall n m (EQ: n = 2 * m), similarity (f @ n) (g @ m).
  Proof.
    i; eapply (@simul_tarski fgsim); [clear n m EQ|by eauto].
    i; destruct PR; subst; try by unfold_rfg; simul_step 1; fold_rfg.
    - by simul_step 0; fold_rfg; eauto.
    - by simul_step 0; fold_rfg; eauto.
    - by destruct m0; simul_step 2; eauto.
    - by destruct m; simul_step 1; eauto.
  Qed.

```

# Proof using Tarski's Principle

```

inductive fgsim : exp -> exp -> Prop :=
  _fgsim1: forall (n m: nat) (EQ: n = 2 * m), fgsim
    (f @ n)
    (g @ m)
  _fgsim2: forall (m: nat), fgsim
    (Let "v" := (Eoutput (m + (m + 0)));; Einput) ~* 2
    in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))) |
    (Eoutput (2 ~* m));;
    (Let "v" := Einput
     in If "v" ~<= 0 then (restart @ g) @ (2 ~* m) else g @ ("v" ~+ m)))
  _fgsim3: forall (m: nat), fgsim
    (Let "v" := (<>);; Einput) ~* 2
    in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))) |
    (<>);;
    (Let "v" := Einput
     in If "v" ~<= 0 then (restart @ g) @ (2 ~* m) else g @ ("v" ~+ m)))
  _fgsim4: forall (m: nat), fgsim
    (Let "v" := Einput ~* 2
     in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))))
    (Let "v" := Einput
     in If "v" ~<= 0 then (restart @ g) @ (2 ~* m) else g @ ("v" ~+ m))
  _fgsim5: forall (m m0: nat), fgsim
    (Let "v" := m0 ~* 2
     in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))))
    (Let "v" := m0
     in If "v" ~<= 0 then (restart @ g) @ (2 ~* m) else g @ ("v" ~+ m))
  _fgsim6: forall (m m0: nat), fgsim
    (Let "v" := m0 * 1 + m0
     in (If 1 ~<= "v" then f else restart @ f) @ ("v" ~+ (m + (m + 0))) |
     (If m0 ~<= 0 then (restart @ g) @ (2 ~* m) else g @ (m0 ~+ m)))
  _fgsim7: forall (m m0: nat), fgsim
    ((If 1 ~<= (m0 * 1 + m0) then f else restart @ f) @
     (m0 * 1 + m0) ~+ (m + (m + 0))) |
    (If ble_nat m0 0 then (restart @ g) @ (2 ~* m) else g @ (m0 ~+ m))
  _fgsim8: forall (m m0: nat), fgsim
    ((If 1 ~<= (m0 * 1 + m0) then f else restart @ f) @
     (m0 * 1 + m0 + (m + (m + 0)))) |
    (If ble_nat m0 0 then (restart @ g) @ (2 ~* m) else g @ (m0 ~+ m))
  _fgsim9: forall (m m0: nat), fgsim
    ((If ble_nat 1 (m0 * 1 + m0) then f else restart @ f) @
     (m0 * 1 + m0 + (m + (m + 0)))) |
    (If ble_nat m0 0 then (restart @ g) @ (2 ~* m) else g @ (m0 ~+ m))
  _fgsim10: forall (m: nat), fgsim
    ((restart @ f) @ m) ((restart @ g) @ m)
  _fgsim11: forall (m: nat), fgsim
    (Efix " " "n"
     (If "n" ~<= 0 then f @ 0
      else Eoutput "n";; (restart @ f) @ ("n" ~- 1)) @ m)
    (Efix " " "n"
     (If "n" ~<= 0 then g @ 0
      else Eoutput "n";; (restart @ g) @ ("n" ~- 1)) @ m)
  _fgsim12: forall (m: nat), fgsim
    (If m ~<= 0 then f @ 0 else Eoutput m;; (restart @ f) @ (m ~- 1))
    (If m ~<= 0 then g @ 0 else Eoutput m;; (restart @ g) @ (m ~- 1))
  _fgsim13: forall (m: nat), fgsim
    (If ble_nat m 0 then f @ 0 else Eoutput m;; (restart @ f) @ (m ~- 1))
    (If ble_nat m 0 then g @ 0 else Eoutput m;; (restart @ g) @ (m ~- 1))
  _fgsim14: forall (m: nat), fgsim
    (Eoutput (S m);; (restart @ f) @ (S m ~- 1))
    (Eoutput (S m);; (restart @ g) @ (S m ~- 1))
  _fgsim15: forall (m: nat), fgsim
    (<>);; (restart @ f) @ (S m ~- 1)
    (<>);; (restart @ g) @ (S m ~- 1)
  _fgsim16: forall (m: nat), fgsim
    ((restart @ f) @ (S m ~- 1))
    ((restart @ g) @ (S m ~- 1))

lemma rsp_simulated_tarski:
  forall n m (EQ: n = 2 * m), similarity (f @ n) (g @ m).
Proof.
  i; eapply (@simul_tarski fgsim); [clear n m EQ|by eauto].
  i; destruct PR; subst; try by unfold_rfg; simul_step 1; fold_rfg.
  - by simul_step 0; fold_rfg; eauto.
  - by simul_step 0; fold_rfg; eauto.
  - by destruct m0; simul_step 2; eauto.
  - by destruct m; simul_step 1; eauto.
  end.

```

# Proof using Guarded Coinduction

```
lemma rsp_simulated_cofix:  
  forall n m (EQ: n = 2 * m), similarity : (f @ n) (g @ m).  
Proof.  
  cofix CIH.  
  intros; subst; do 6 csimul_step 1; do 2 csimul_step 0.  
  destruct m0; csimul_step 2; [|by eauto|].  
  fold_rfg; generalize (m+(m+0)).  
  cofix CIH'.  
  intros; do 3 csimul_step 1.  
  destruct n; csimul_step 1; [by eauto|].  
  do 3 csimul_step 1; eauto.  
  id.
```

# Proof using Guarded Coinduction

Thanks to

Incremental Proof + Simple Automation

```
lemma rsp_simulated_cofix:  
  forall n m (EQ: n = 2 * m), similarity : (f @ n) (g @ m).  
Proof.  
  cofix CIH.  
  intros; subst; do 6 csimul_step 1; do 2 csimul_step 0.  
  destruct m0; csimul_step 2; [|by eauto|].  
  fold_rfg; generalize (m+(m+0)).  
  cofix CIH'.  
  intros; do 3 csimul_step 1.  
  destruct n; csimul_step 1; [by eauto|].  
  do 3 csimul_step 1; eauto.  
  id.
```

# Proof using Guarded Coinduction

Thanks to

Incremental Proof + Simple Automation

**BUT!!!**

```
lemma rsp_simulated_cofix:
  forall n m (EQ: n = 2 * m), similarity : (f @ n) (g @ m).
Proof.
  cofix CIH.
  intros; subst; do 6 csimul_step 1; do 2 csimul_step 0.
  destruct m0; csimul_step 2; [|by eauto|].
  fold_rfg; generalize (m+(m+0)).
  cofix CIH'.
  intros; do 3 csimul_step 1.
  destruct n; csimul_step 1; [by eauto|].
  do 3 csimul_step 1; eauto.
  id.
```

# Problems with Syntactic Guardedness

```
Variable Node: Type.
Variable R: relation Node.

CoInductive inpath x : Prop :=
  _inpath y (STEP: R x y) (INF: inpath y).
Hint Constructors inpath.

Lemma Park's_principle:
  forall (P: Node -> Prop),
    (forall x, P x -> exists y, clos_trans_in _ R x y /\ P y) ->
    forall x, P x -> inpath x.
Proof.
  cofix CIH; intros P M x Px.
  destruct (M _ Px) as (y & C & Py); clear Px.
  induction C; eauto.
Qed.
```

```
-\--- x.v          56% (51,0)      (Coq Script(0-) Holes)
```

```
No more subgoals.
```

```
(dependent evars: ?250 using , ?260 using , ?270 using ,)
```

# Problems with Syntactic Guardedness

```
Variable Node: Type.  
Variable R: relation Node.
```

```
CoInductive inpath x : Prop :=  
  _inpath y (STEP: R x y) (INF: inpath y).  
Hint Constructors inpath.
```

```
Lemma Park's_principle:
```

```
  forall (P: Node -> Prop),  
  (forall x, P x -> exists y, clos_trans_in _ R x y /\ P y) ->  
  forall x, P x -> inpath x.
```

```
Proof.
```

```
  cofix CIH; intros P M x Px.  
  destruct (M _ Px) as (y & C & Py); clear Px.  
  induction C; eauto.
```

```
Qed.
```

```
-\--- x.v          56% (51,0)      (Coq Script(0-) Holes)
```

```
No more subgoals.
```

```
(dependent evars: ?250 using , ?260 using , ?270 using ,)
```



# Problems with Syntactic Guardedness

```
Variable Node: Type.
Variable R: relation Node.

CoInductive infpath x : Prop :=
  _infpath y (STEP: R x y) (INF: infpath y).
Hint Constructors infpath.
```

```
Lemma Park's_principle:
  forall (P: Node -> Prop),
    (forall x, P x -> exists y, clos_trans_in _ R x y /\ P y) ->
    forall x, P x -> infpath x.
```

```
PROOF.
  cofix CIH; intros P M x Px.
  destruct (M _ Px) as (y & C & Py); clear Px.
  induction C; eauto.
Qed.
```

```
- \--- x.v          56% (51,0)      (Coq Script(0-) Holes)
```

```
No more subgoals.
```

```
(dependent evars: ?250 using , ?260 using , ?270 using ,)
```

# Problems with Syntactic Guardedness

```
Variable Node: Type.
Variable R: relation Node.

CoInductive inpath x : Prop :=
  _inpath y (STEP: R x y) (INF: inpath y).
Hint Constructors inpath.

Lemma Park's_principle:
  forall (P: Node -> Prop),
  (forall x, P x -> exists y, clos_trans_in _ R x y /\ P y) ->
  forall x, P x -> inpath x.
Proof
  cofix CIH intros P M x Px.
  destruct (M _ Px) as (y & C & Py); clear Px.
  induction C; eauto.
Qed.
```

```
- \--- x.v          56% (51,0)      (Coq Script(0-) Holes)
```

```
No more subgoals.
```

```
(dependent evars: ?250 using , ?260 using , ?270 using ,)
```

# Problems with Syntactic Guardedness

```
Variable Node: Type.
Variable R: relation Node.

CoInductive infpath x : Prop :=
  _infpath y (STEP: R x y) (INF: infpath y).
Hint Constructors infpath.

Lemma Park's_principle:
  forall (P: Node -> Prop),
  (forall x, P x -> exists y, clos_trans_in _ R x y /\ P y) ->
  forall x, P x -> infpath x.
Proof.
  cofix CIH; intros P M x Px.
  destruct (M _ Px) as (y & C & Py); clear Px.
  induction C; eauto.
Qed.
```

```
- \--- x.v          56% (51,0)      (Coq Script(0-) Holes)
```

```
No more subgoals.
```

```
(dependent evars: ?250 using , ?260 using , ?270 using ,)
```

# Problems with Syntactic Guardedness

```
Variable Node: Type.
Variable R: relation Node.

CoInductive inpath x : Prop :=
  _inpath y (STEP: R x y) (INF: inpath y).
Hint Co

Lemma P
  forall
  (fora
  forall

Proof.
  cofix CIH; intros P M x Px
  destruct (M _ Px) as (y C & Py); clear Px.
  induction C; eauto.

Qed.
□
```

The proof is  
**NOT Syntactically** Guarded

```
- \--- x.v 56% (52,0) (Coq Script(0-) Holes)
Error:
Recursive definition of CIH is ill-formed.
```

# Problems with Syntactic Guardedness

## Guardedness NOT Expressed in the Logic

1. Far from complete
2. Bad interaction with automation
3. Hard to debug
4. Slow

```
Proof.
```

```
cofix CIH; intros P M x Px.  
destruct (M _ Px) as (y & C & Py); clear Px.  
induction C; eauto.
```

```
Qed.
```

```
□
```

```
- \--- x.v 56% (52,0) (Coq Script(0-) Holes)
```

```
Error:
```

```
Recursive definition of CIH is ill-formed.
```

# Problems with Syntactic Guardedness

## Guardedness NOT Expressed in the Logic

1. Far from complete
2. Bad interaction with automation
3. Hard to debug
4. Slow

```
Proof.
  cofix CIH; intros P M x Px.
  destruct (M _ Px) as (y & C & Py); clear Px.
  induction C; auto.
Qed.
□
- \--- x.v          56% (52,0)      (Coq Script(0-) Holes)
Error:
Recursive definition of CIH is ill-formed.
```

# Summary:

## Pros and Cons of Two Principles

- Tarski's Fixed Point Theorem
  - + Simple & Robust
  - Inconvenient to use
- Syntactically Guarded Coinduction
  - Complex & Fragile due to "Guardedness Checking"
  - + More Convenient to use

# Talk Outline

## □ Previous Approaches

- Tarski's Fixed Point Theorem
- Syntactically Guarded Coinduction

## □ Our Approach



- Parameterized Coinduction



# Syntactically Guarded Coinduction

$$f : \wp(U) \xrightarrow{\text{mon}} \wp(U)$$

Syntactically Guarded Coinduction

$$\frac{\text{pf: } X \subseteq \nu f \implies X \subseteq \nu f}{X \subseteq \nu f} \quad (\text{pf is guarded})$$

# Semantically Guarded Coinduction

Guardedness Expressed **Directly Within** the Logic

Semantically Guarded Coinduction

$$X \subseteq \nu f \stackrel{G}{\Longrightarrow} X \subseteq \nu f$$

---

$$X \subseteq \nu f$$

# Guarded Implication

$$Y \subseteq \nu f \stackrel{G}{\Rightarrow} X \subseteq \nu f$$

# Guarded Implication

Do **NOT** want to **EXTEND** the logic

$$Y \subseteq \nu f \stackrel{G}{\Rightarrow} X \subseteq \nu f$$

# Guarded Implication

Instead, Define  $G_f$   
Called “Parameterized Greatest Fixed Point” of  $f$

$$Y \subseteq \nu f \stackrel{G}{\Rightarrow} X \subseteq \nu f$$



$$X \subseteq G_f(Y)$$

# Guarded Implication

Greatest Fixed Point of  $f$   
Under Guarded Assumption of  $Y \subseteq \nu f$

$$Y \subseteq \nu f \xRightarrow{G} X \subseteq \nu f$$

$$X \subseteq G_f(Y)$$

# Guarded Implication

Greatest Fixed Point of  $f$   
Under Guarded Assumption of  $Y \subseteq \nu f$

$$Y \subseteq \nu f \xRightarrow{G} X \subseteq \nu f$$

$$X \subseteq G_f(Y)$$

$$G_f(Y) \stackrel{\text{def}}{=} \nu(\lambda S. f(Y \cup S))$$

# Properties of $G_f$

(Init)  $G_f(\emptyset) = \nu f$

(Unfolding)  $G_f(Y) = f(Y \cup G_f(Y))$

(Semantically Guarded Coinduction)

$$X \subseteq G_f(Y) \iff X \subseteq G_f(X \cup Y)$$



# Properties of $G_f$

(Init)  $G_f(\emptyset) = \nu f$

(Unfolding)  $G_f(Y) = f(Y \cup G_f(Y))$

(Semantically Guarded Coinduction)

$$X \subseteq G_f(Y) \iff X \subseteq G_f(X \cup Y)$$

# Properties of $G_f$

(Init)  $G_f(\emptyset) = \nu f$

(Unfolding)  $G_f(Y) = f(Y \cup G_f(Y))$

(Semantically Guarded Coinduction)

$$X \subseteq G_f(Y) \iff X \subseteq G_f(X \cup Y)$$

# Properties of $G_f$

(Init)  $G_f(\emptyset) = \nu f$

(Unfolding)  $G_f(Y) = f(Y \cup G_f(Y))$

(Semantically Guarded Coinduction)

$$X \subseteq G_f(Y) \iff X \subseteq G_f(X \cup Y)$$

# Properties of $G_f$

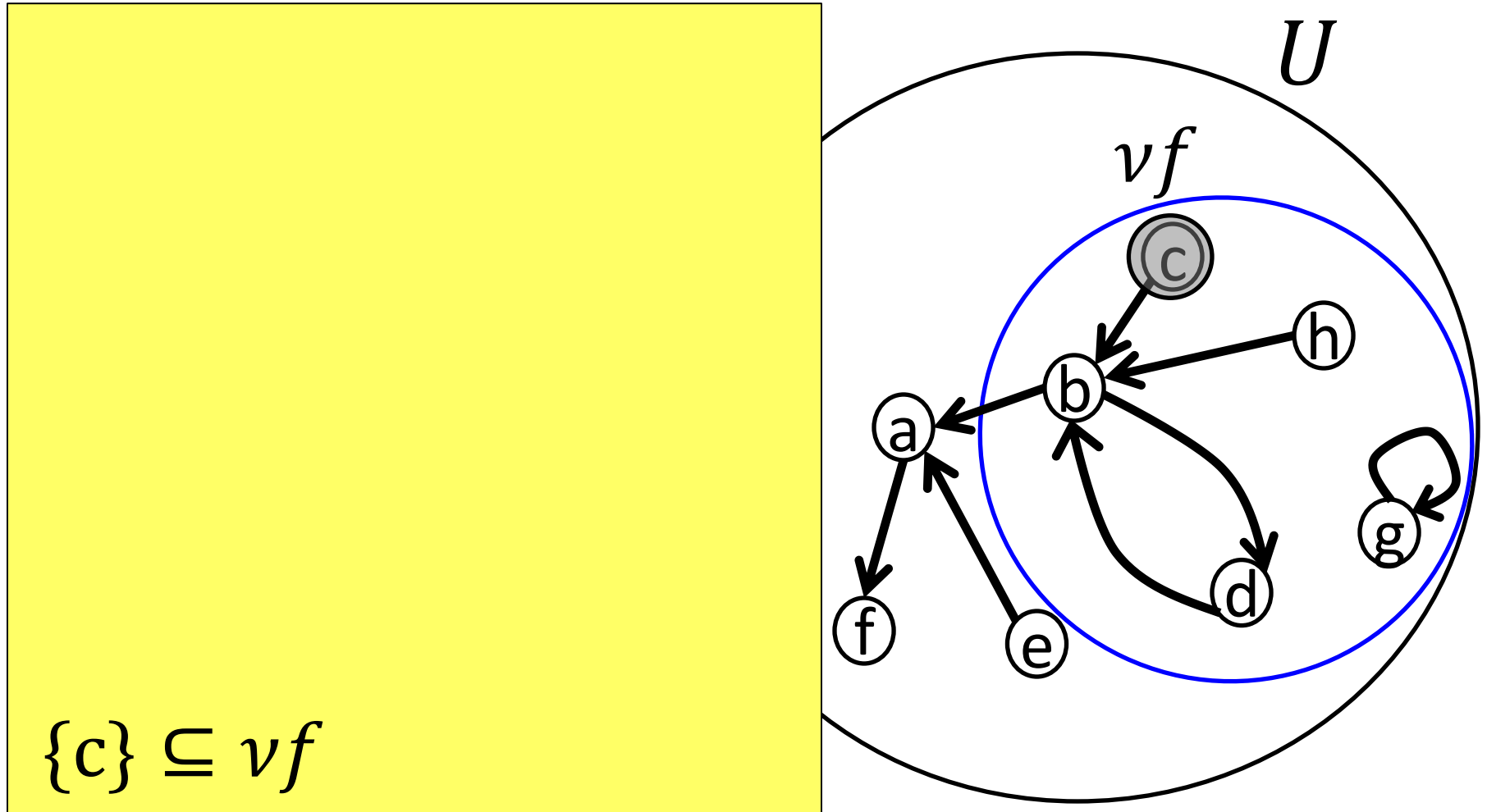
(Init)  $G_f(\emptyset) = \nu f$

(Unfolding)  $G_f(Y) = f(Y \cup G_f(Y))$

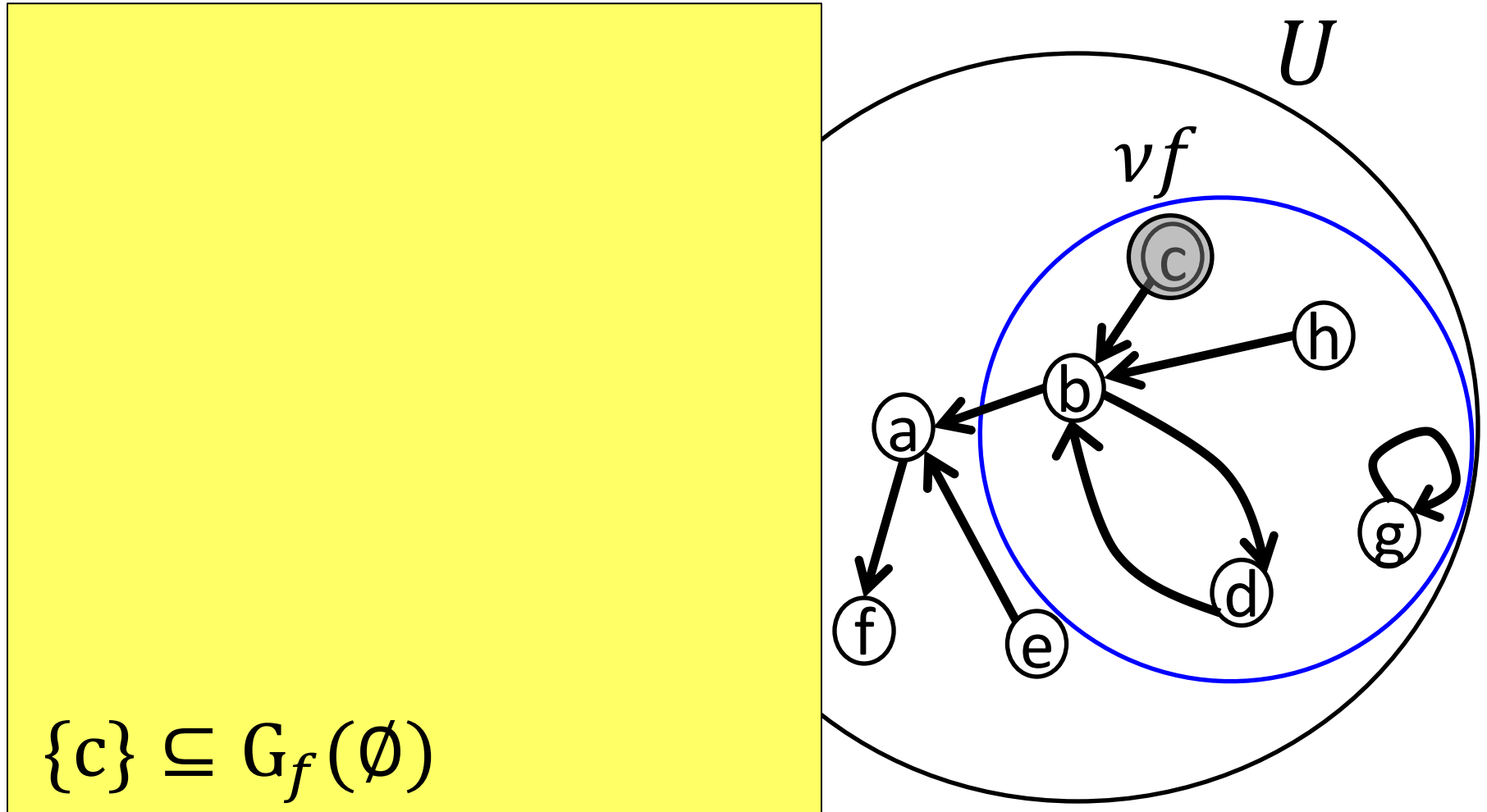
(Semantically Guarded Coinduction)

$$X \subseteq G_f(Y) \iff X \subseteq G_f(X \cup Y)$$

# Example: Parameterized Coinduction

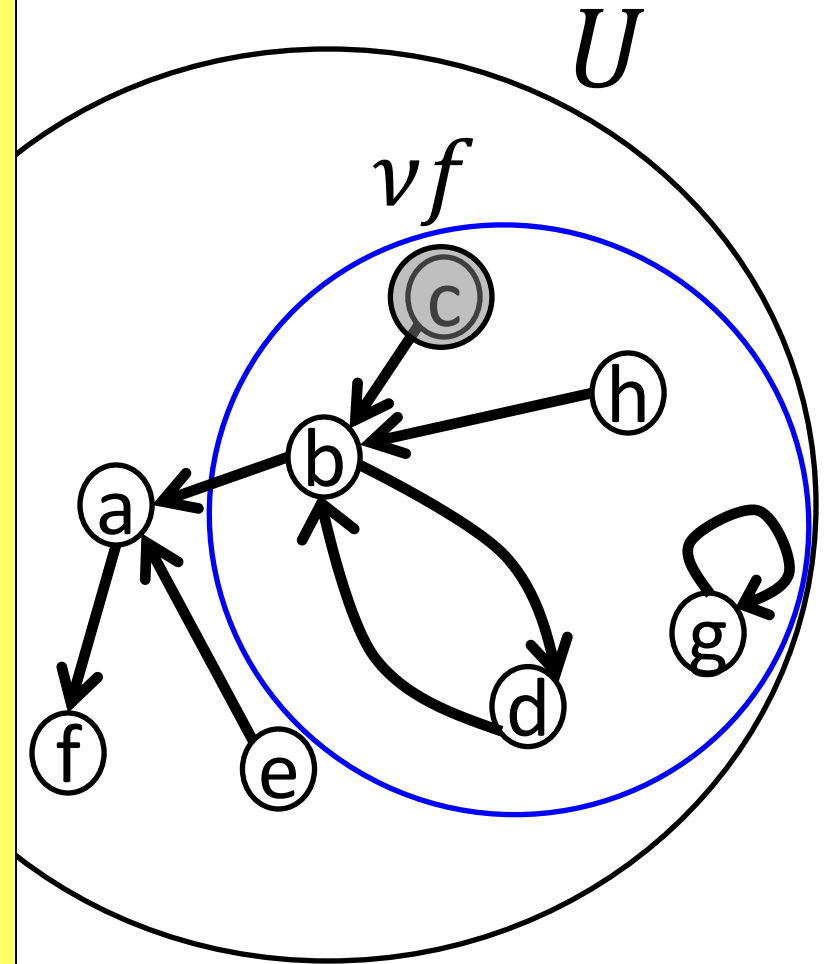


# Example: Parameterized Coinduction



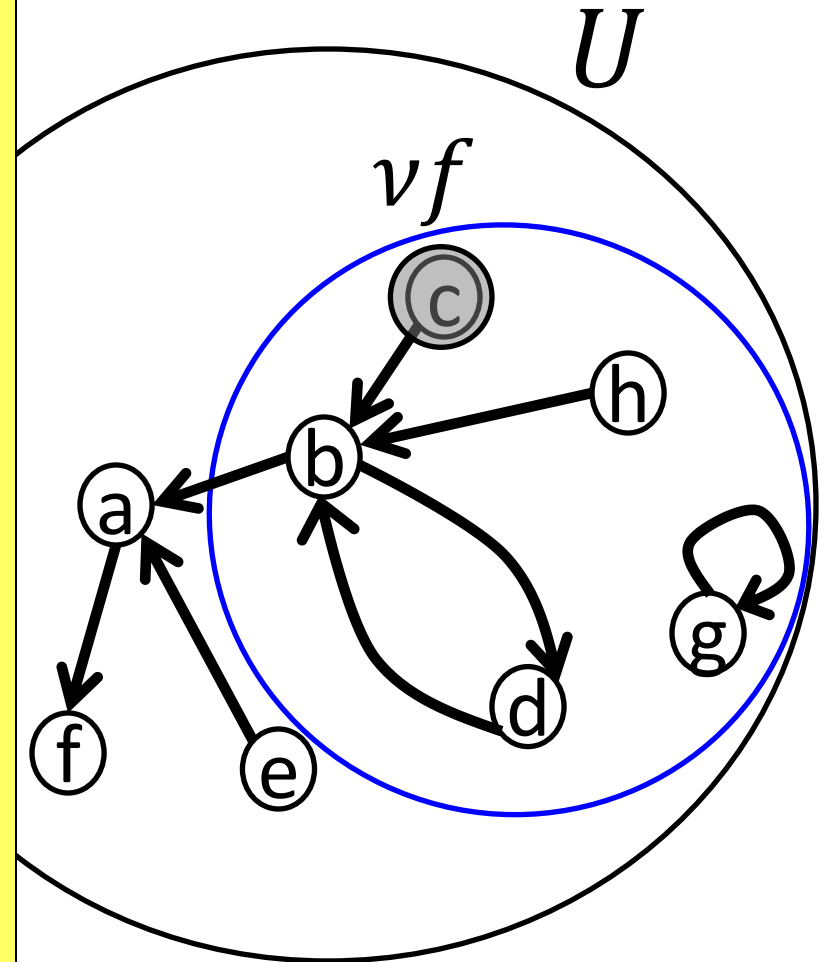
# Example: Parameterized Coinduction

$$\frac{\{c\} \subseteq f(\emptyset \cup G_f(\emptyset))}{\{c\} \subseteq G_f(\emptyset)}$$



# Example: Parameterized Coinduction

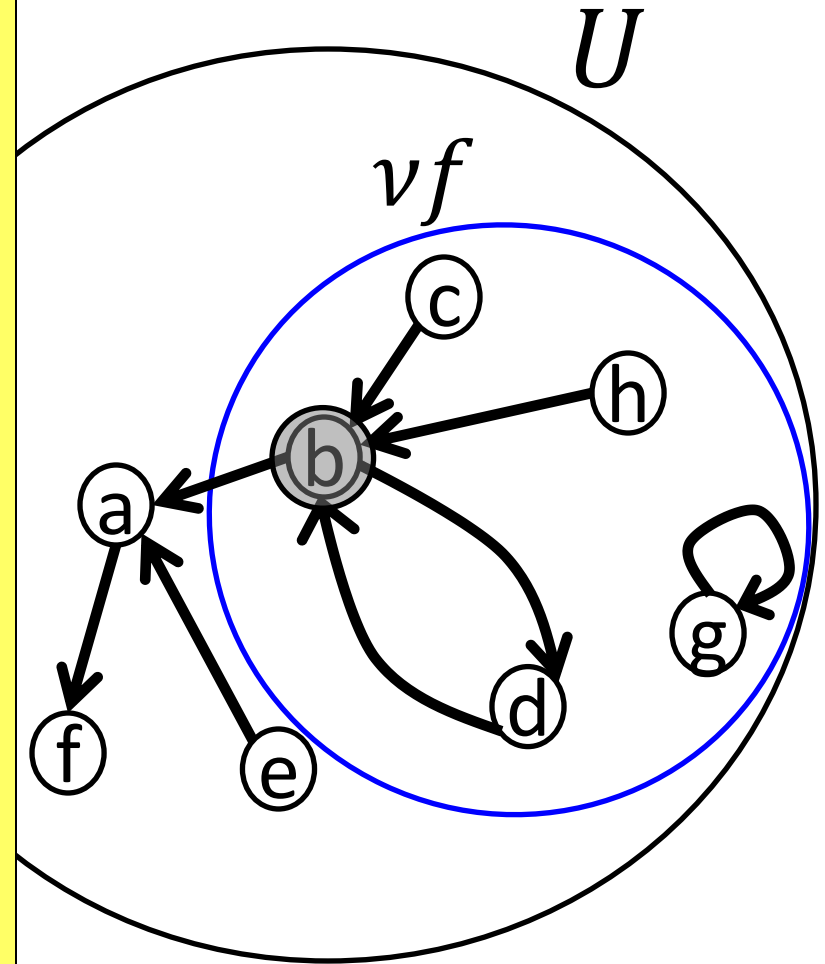
$$\frac{\exists y. c \rightarrow y \wedge y \in (\emptyset \cup G_f(\emptyset))}{\{c\} \subseteq G_f(\emptyset)}$$





# Example: Parameterized Coinduction

$$\frac{\exists y. c \rightarrow y \wedge y \in (\emptyset \cup G_f(\emptyset))}{\{c\} \subseteq G_f(\emptyset)}$$

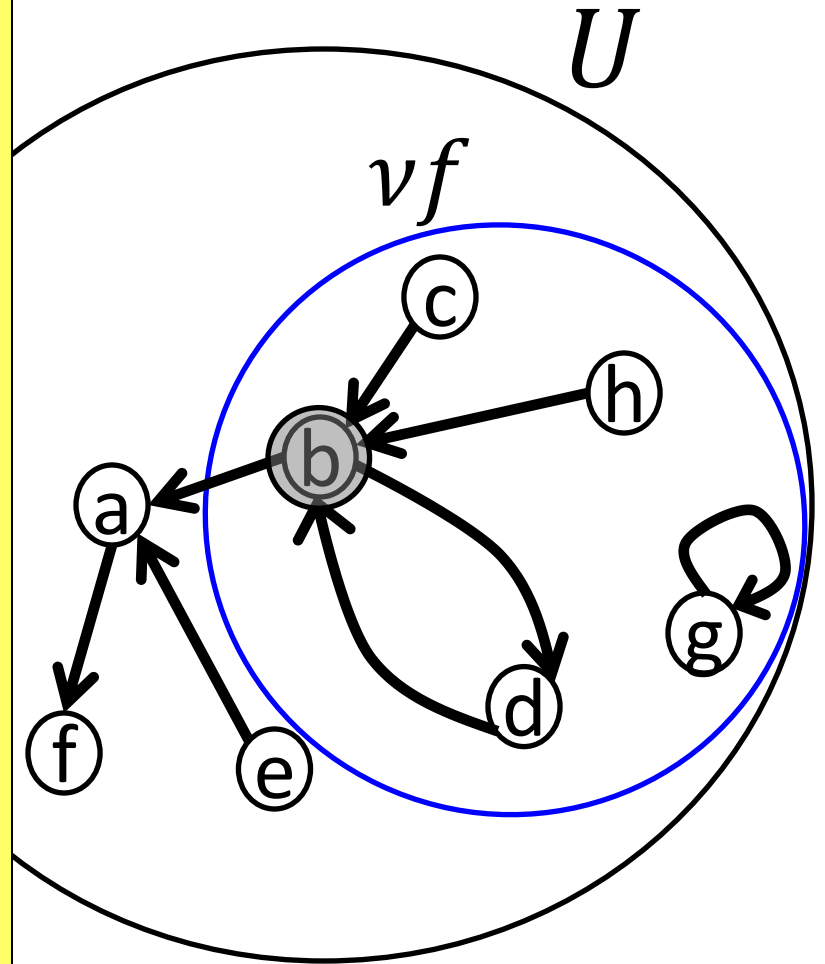


# Example: Parameterized Coinduction

$b \in (\emptyset \cup G_f(\emptyset))$

---

$\{c\} \subseteq G_f(\emptyset)$



# Example: Parameterized Coinduction

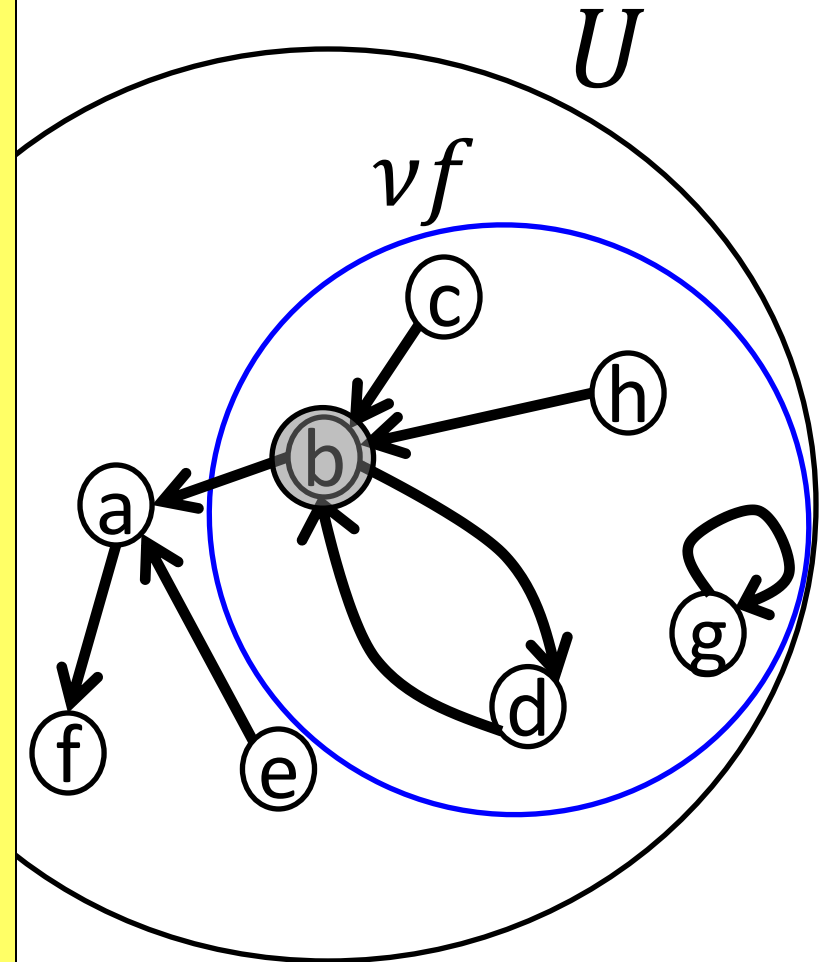
$$\{b\} \subseteq G_f(\emptyset)$$

---

$$b \in (\emptyset \cup G_f(\emptyset))$$

---

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

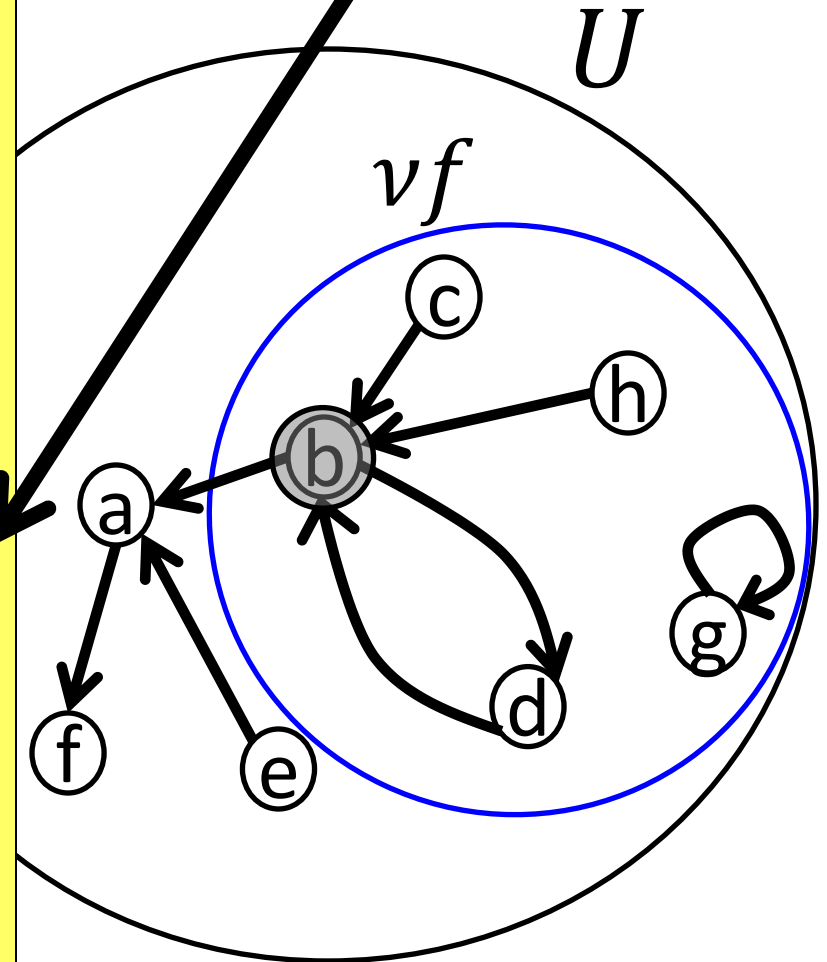
By **Semantically  
Guarded Coinduction**

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

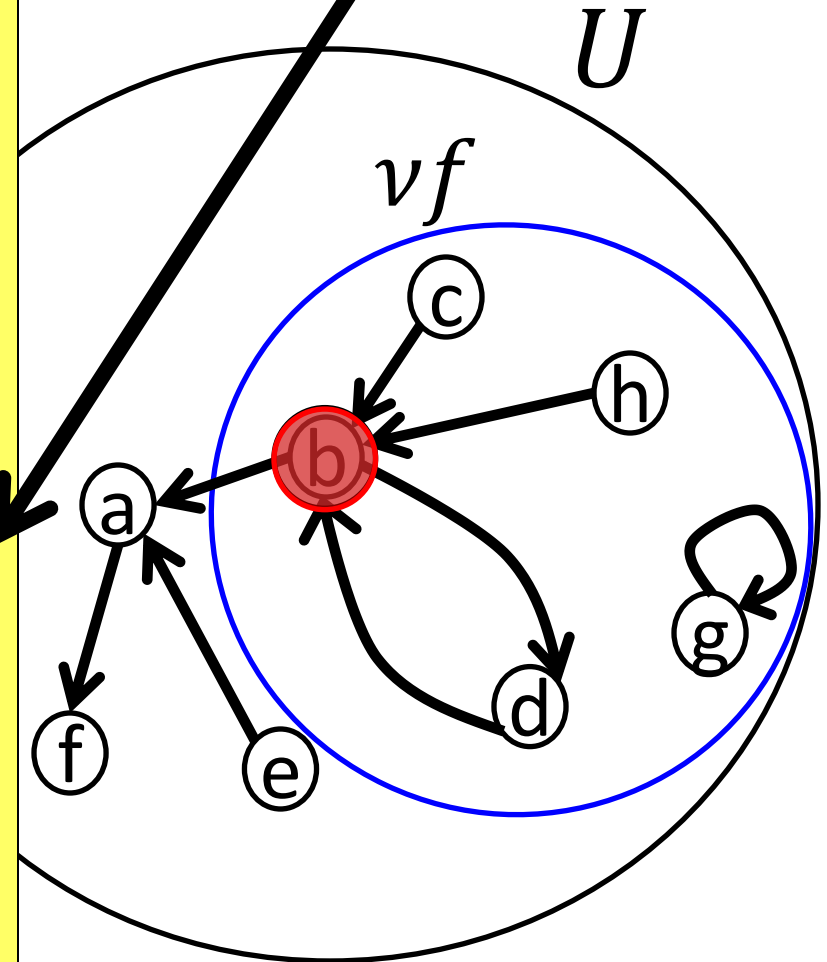
By **Semantically  
Guarded Coinduction**

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$



NO Trivial Proof

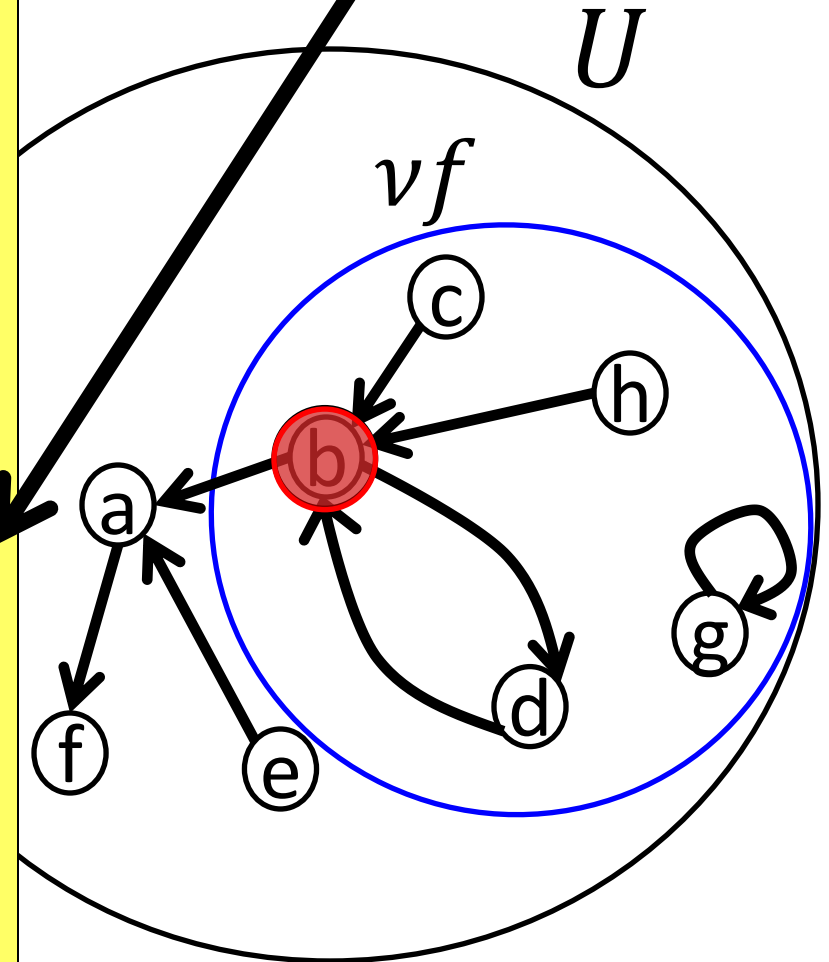
By **Semantically**  
Guarded Coinduction

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized

By **Semantically  
Guarded Coinduction**

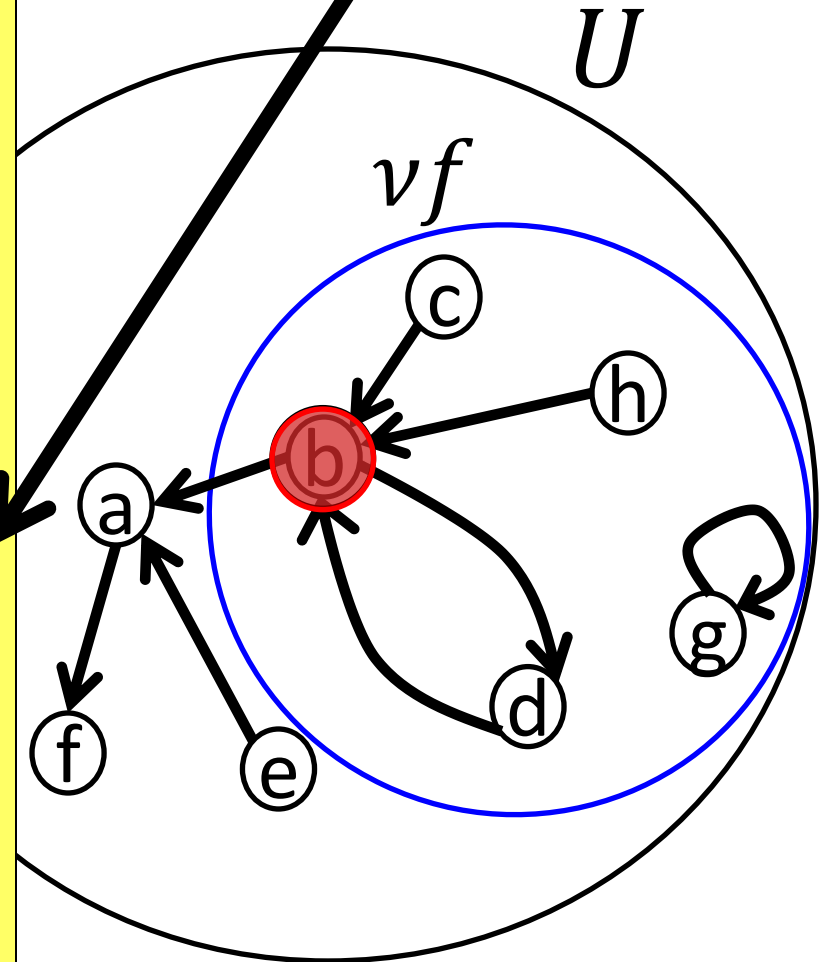
$$\{b\} \subseteq f(\{b\} \cup G_f(\{b\}))$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

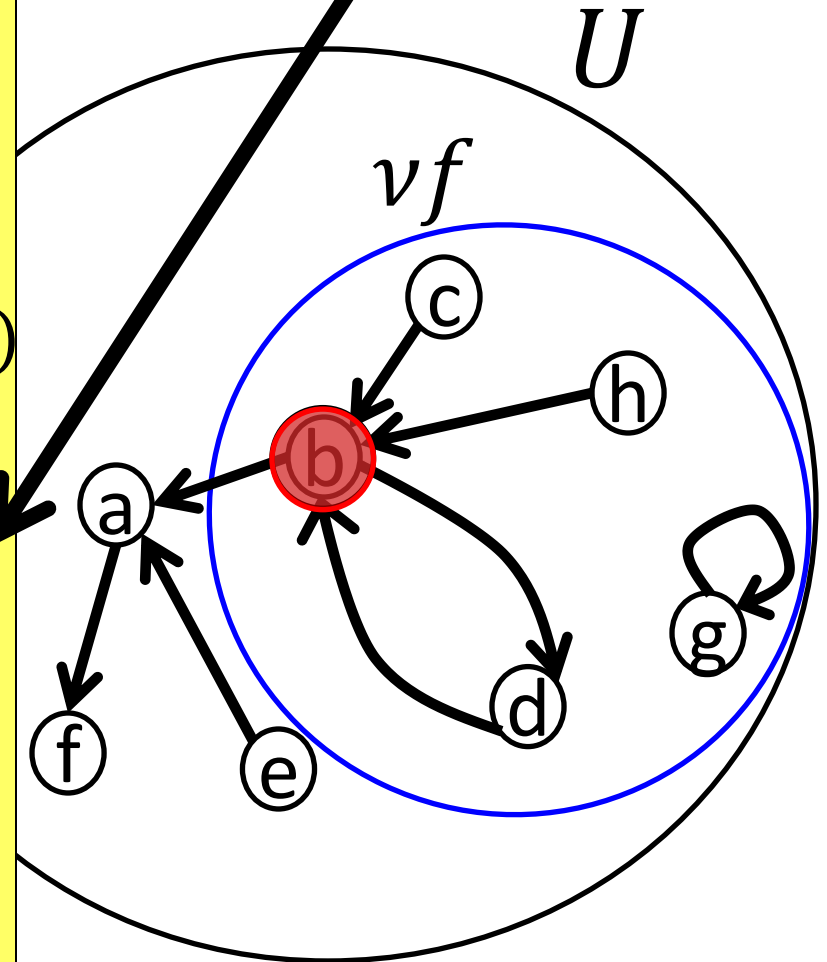
$$\frac{\exists y. b \rightarrow y \wedge y \in (\{b\} \cup G_f(\{b\}))}{\{b\} \subseteq G_f(\{b\})}$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$





# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

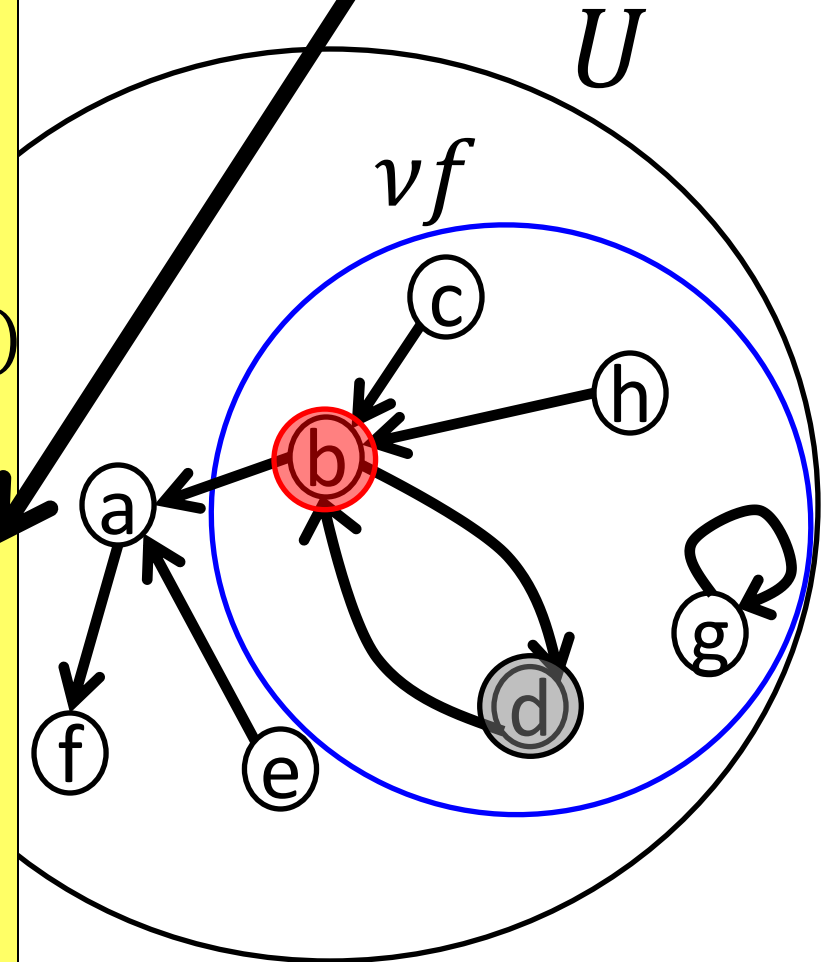
$$\frac{\exists y. b \rightarrow y \wedge y \in (\{b\} \cup G_f(\{b\}))}{\{b\} \subseteq G_f(\{b\})}$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

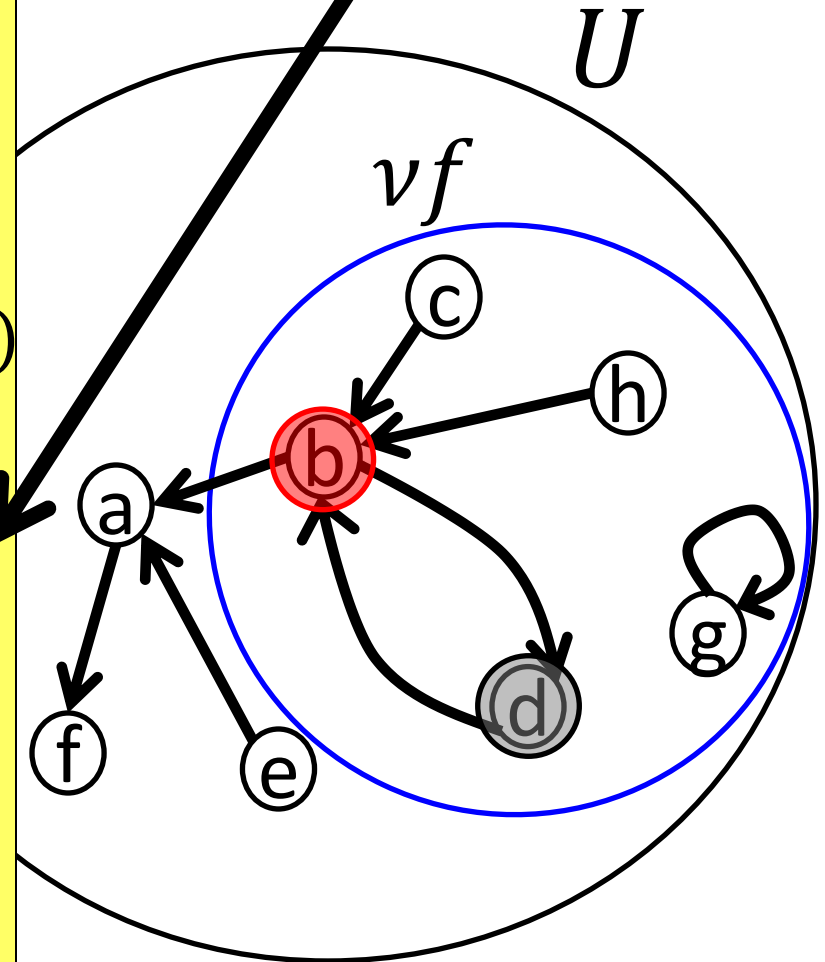
$$d \in (\{b\} \cup G_f(\{b\}))$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

$$\{d\} \subseteq G_f(\{b\})$$

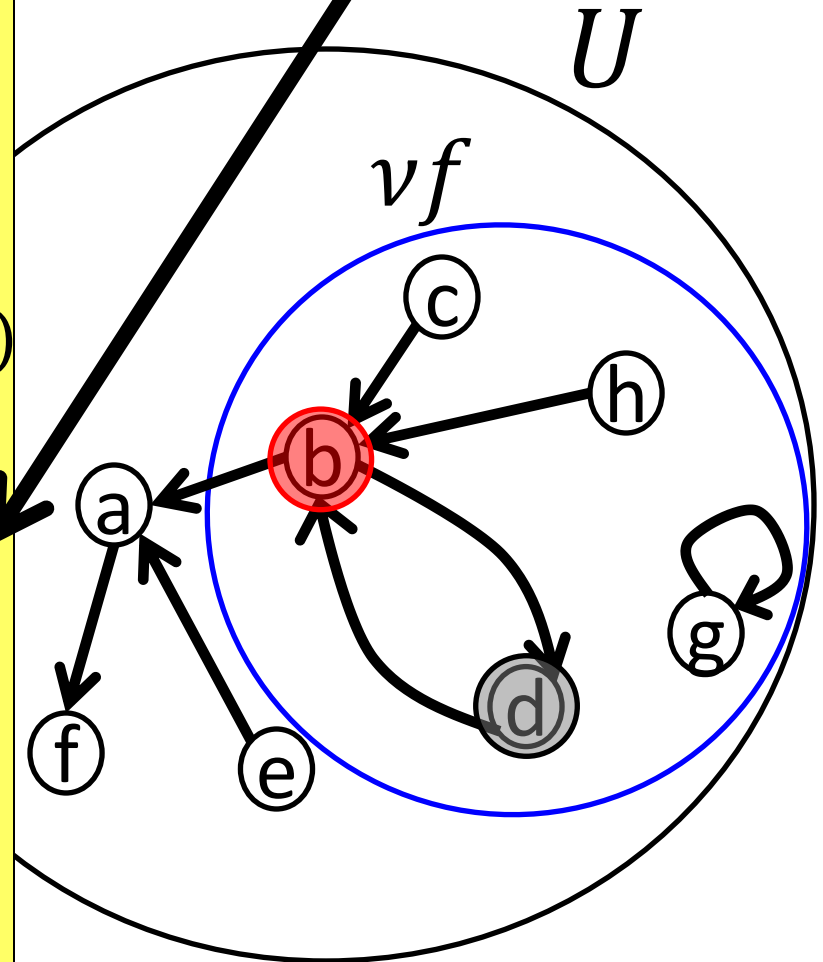
$$d \in (\{b\} \cup G_f(\{b\}))$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

$$\{d\} \subseteq f(\{b\} \cup G_f(\{b\}))$$

$$\{d\} \subseteq G_f(\{b\})$$

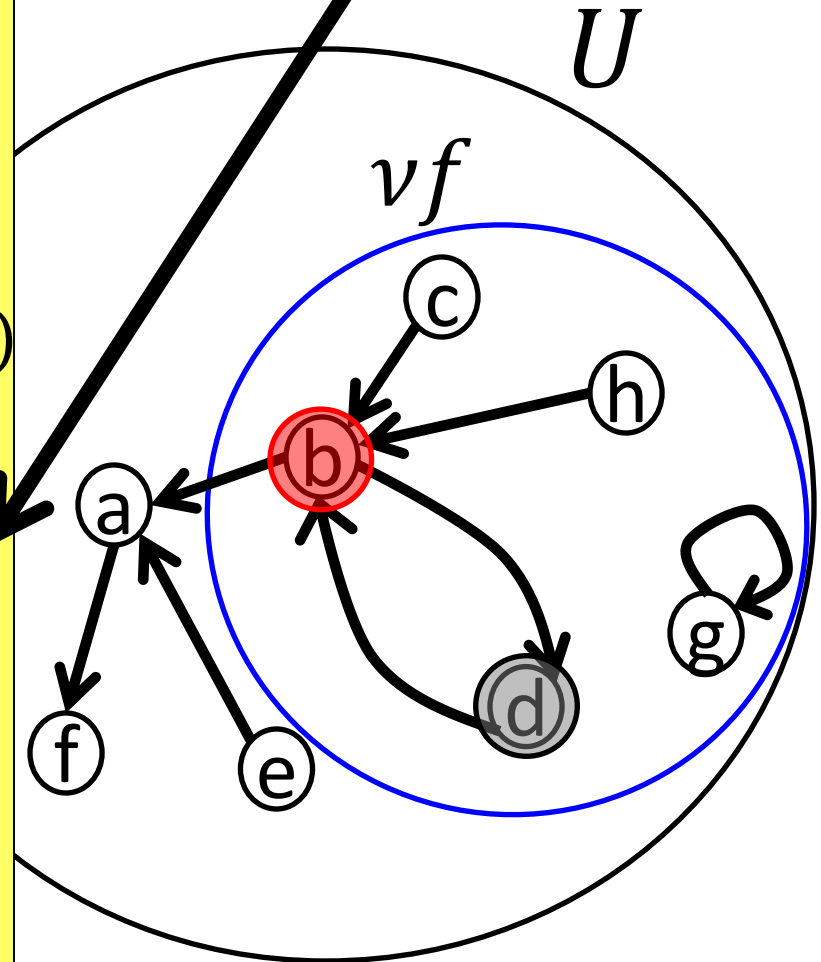
$$d \in (\{b\} \cup G_f(\{b\}))$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

$$\frac{\exists y. d \rightarrow y \wedge y \in (\{b\} \cup G_f(\{b\}))}{\{d\} \subseteq G_f(\{b\})}$$

$$\{d\} \subseteq G_f(\{b\})$$

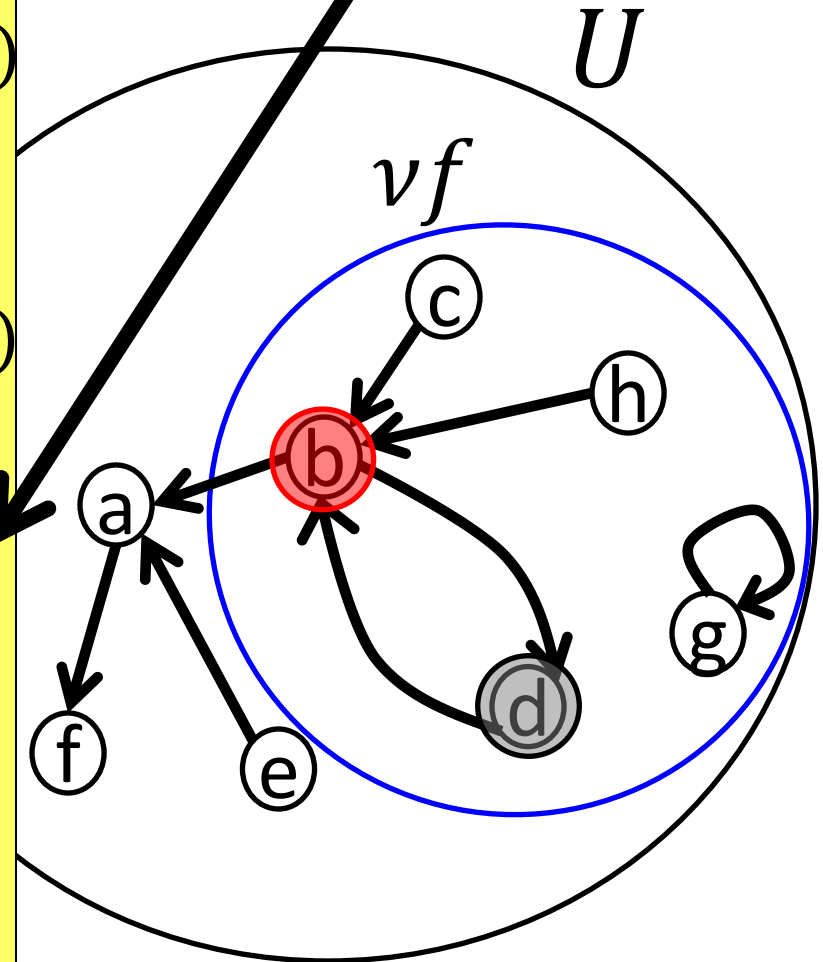
$$\frac{d \in (\{b\} \cup G_f(\{b\}))}{\{b\} \subseteq G_f(\{b\})}$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$\frac{b \in (\emptyset \cup G_f(\emptyset))}{\{c\} \subseteq G_f(\emptyset)}$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

$$\frac{\exists y. d \rightarrow y \wedge y \in (\{b\} \cup G_f(\{b\}))}{\{d\} \subseteq G_f(\{b\})}$$

$$\{d\} \subseteq G_f(\{b\})$$

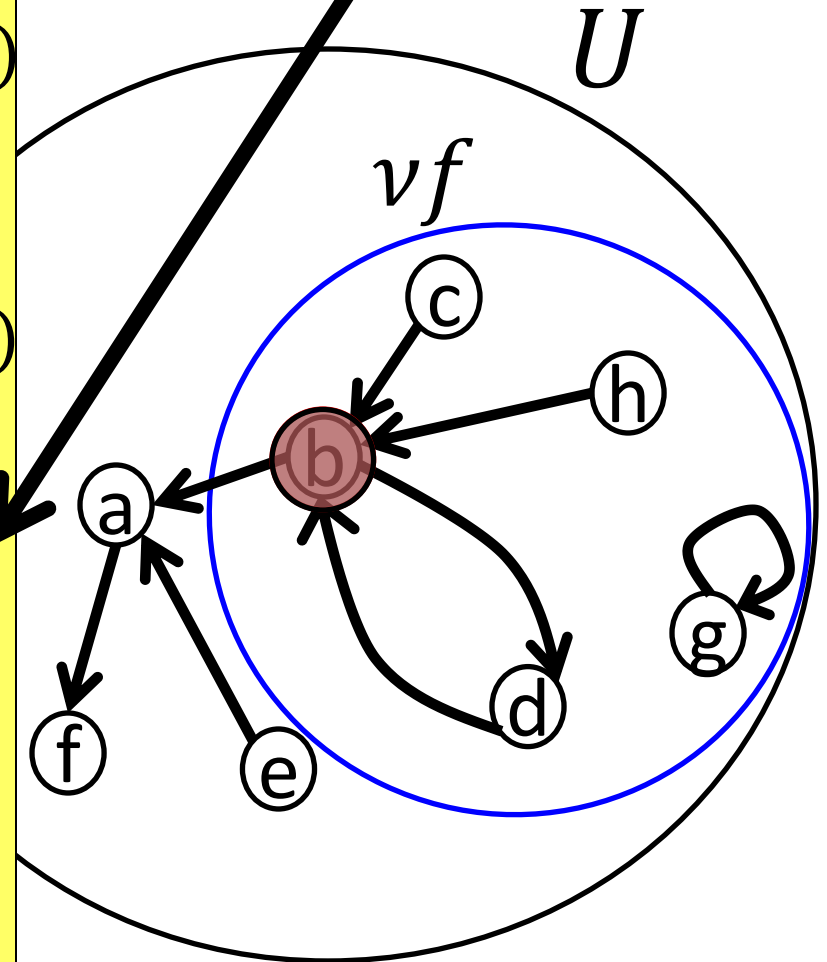
$$\frac{d \in (\{b\} \cup G_f(\{b\}))}{\{b\} \subseteq G_f(\{b\})}$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$\frac{b \in (\emptyset \cup G_f(\emptyset))}{\{c\} \subseteq G_f(\emptyset)}$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

$$b \in (\{b\} \cup G_f(\{b\}))$$

$$\{d\} \subseteq G_f(\{b\})$$

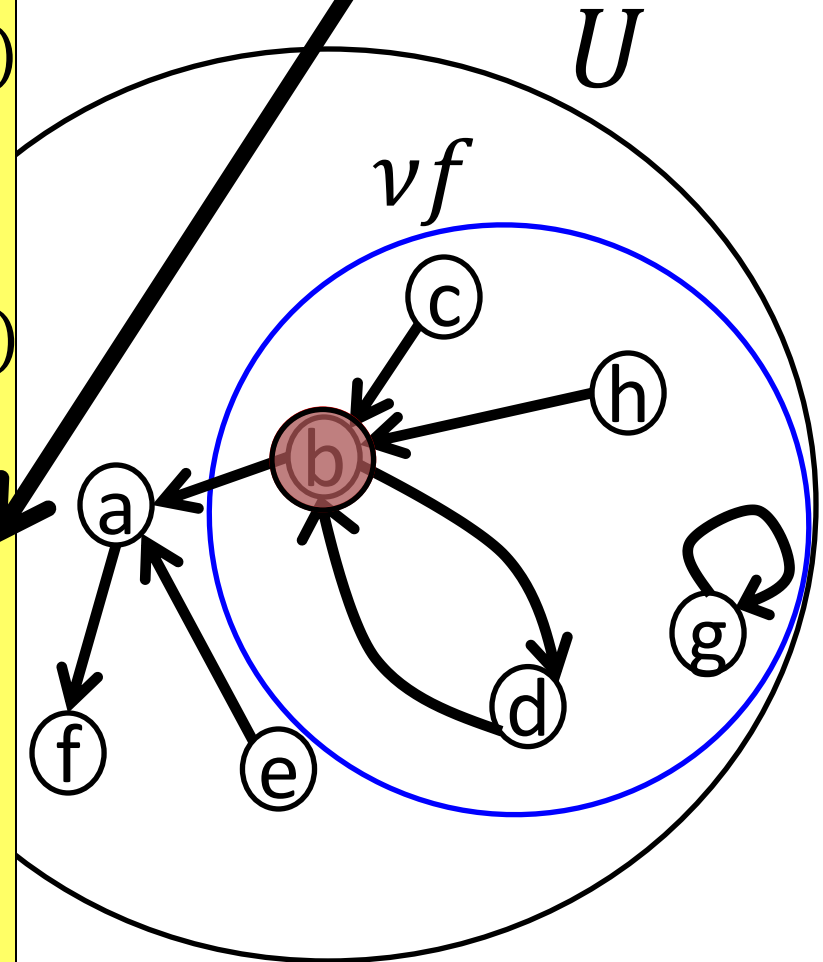
$$d \in (\{b\} \cup G_f(\{b\}))$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$



# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

$$b \in (\{b\} \cup G_f(\{b\}))$$

$$\{d\} \subseteq G_f(\{b\})$$

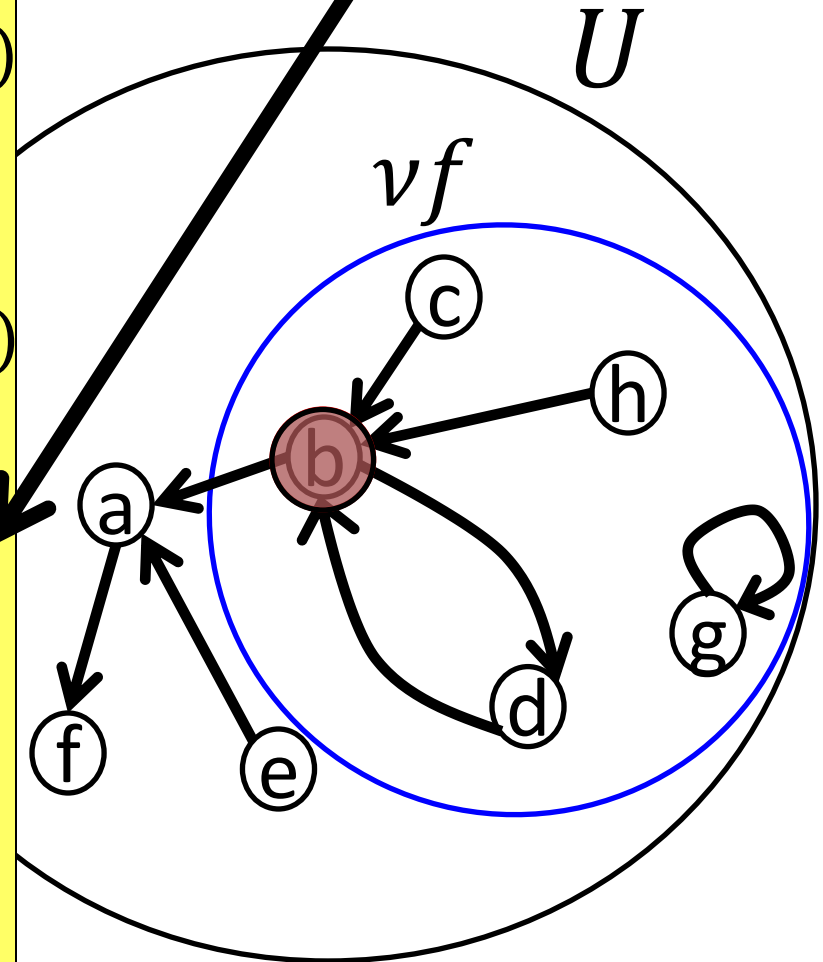
$$d \in (\{b\} \cup G_f(\{b\}))$$

$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$





# Example Parameterized Coinduction

By **Semantically  
Guarded Coinduction**

$$b \in (\{b\} \cup G_f(\{b\}))$$

$$\{d\} \subseteq G_f(\{b\})$$

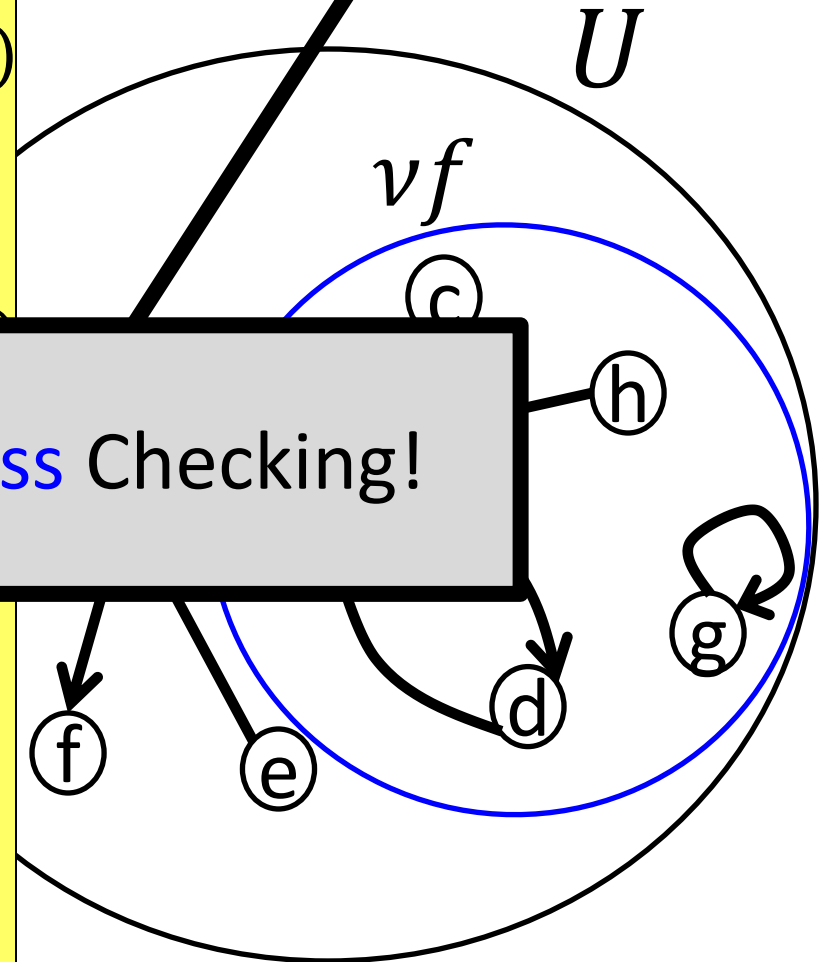
$$\{b\} \subseteq G_f(\{b\})$$

$$\{b\} \subseteq G_f(\emptyset)$$

$$b \in (\emptyset \cup G_f(\emptyset))$$

$$\{c\} \subseteq G_f(\emptyset)$$

**NO Guardedness Checking!**



# Paco: A Coq Library for Parameterized Coinduction

<http://plv.mpi-sws.org/paco/>

**Syntactic** Guardedness

Lemma `rsp_simulated_cofix`:

```
forall n m (EQ: n = 2 * m), similarity (f @ n) (g @ m).
```

Proof.

```
cofix CIH.
```

```
intros; subst; do 6 csimul_step 1; do 2 csimul_step 0.
```

```
destruct m0; csimul_step 2; [|by eauto].
```

```
fold_rfg; generalize (m+(m+0)).
```

```
cofix CIH'.
```

```
intros; do 3 csimul_step 1.
```

```
destruct n; csimul_step 1; [by eauto|].
```

```
do 3 csimul_step 1; eauto.
```

Qed.

**Semantic** Guardedness

Lemma `rsp_simulated_paco`:

```
forall n m (EQ: n = 2 * m), paco2 sim bot2 (f @ n) (g @ m).
```

Proof.

```
pcofix CIH.
```

```
intros; subst; do 6 psimil_step 1; do 2 psimil_step 0.
```

```
destruct m0; psimil_step 2; [|by eauto].
```

```
left; fold_rfg; generalize (m+(m+0)).
```

```
pcofix CIH'.
```

```
intros; do 3 psimil_step 1.
```

```
destruct n; psimil_step 1; [by eauto|].
```

```
do 3 psimil_step 1; eauto.
```

Qed.

# Paco: A Coq Library for Parameterized Coinduction

<http://plv.mpi-sws.org/paco/>

**Syntactic** Guardedness

Lemma `rsp_simulated_cofix`:

```
forall n m (EQ: n = 2 * m), similarity (f @ n) (g @ m).
```

Proof.

```
cofix CIH.
```

```
intros; subst; do 6 csimul_step 1; do 2 csimul_step 0.
```

```
destruct m0; csimul_step 2; [|by eauto].
```

```
fold rfg; generalize (m+(m+0)).
```

```
cofix CIH'.
```

```
intros; do 3 csimul_step 1.
```

```
destruct n; csimul_step 1; [by eauto|].
```

```
do 3 csimul_step 1; eauto.
```

Qed.

**Semantic** Guardedness

Lemma `rsp_simulated_paco`:

```
forall n m (EQ: n = 2 * m), paco2 sim bot2 (f @ n) (g @ m).
```

Proof.

```
pcofix CIH.
```

```
intros; subst; do 6 psimil_step 1; do 2 psimil_step 0.
```

```
destruct m0; psimil_step 2; [|by eauto].
```

```
left; fold_rfg; generalize (m+(m+0)).
```

```
pcofix CIH'.
```

```
intros; do 3 psimil_step 1.
```

```
destruct n; psimil_step 1; [by eauto|].
```

```
do 3 psimil_step 1; eauto.
```

Qed.

# Paco: A Coq Library for Parameterized Coinduction

<http://plv.mpi-sws.org/paco/>

Guardedness Checking

**Needed!**

Stochastic Guardedness

Proof:

```
n = 2 * m), paco2 sim bot2 (f @ n) (g @ m).
```

Proof.

**cofix** CIH.

```
intros; subst; do 6 csimul_step 1; do 2 csimul_step 2; [|by eauto|].  
fold rfg; generalize (m+(m+0)).
```

**cofix** CIH'.

```
intros; do 3 csimul_step 1.  
destruct n; csimul_step 1; [by eauto|].  
do 4 csimul_step 1; eauto.
```

Qed.

Guardedness Checking

**NOT Needed!**

Proof.

intros; do 3 psimil\_step 1.

destruct n; psimil\_step 1; [by eauto|].

do 4 psimil\_step 1; eauto.

Qed.

# Failed Proof using Syntactically Guarded Coinduction

```
Variable Node: Type.
Variable R: relation Node.

CoInductive infpath x : Prop :=
  _infpath y (STEP: R x y) (INF: infpath y).
Hint Constructors infpath.

Lemma Park's_principle:
  forall (P: Node -> Prop),
    (forall x, P x -> exists y, clos_trans_1n _ R x y /\ P y) ->
    forall x, P x -> infpath x.
Proof
  cofix CIH; intros P M x Px.
  destruct (M _ Px) as (y & C & Py); clear Px.
  induction C; eauto.
Qed.
□
- \--- x.v          56% (52,0)      (Coq Script(0-) Holes)
Error:
Recursive definition of CIH is ill-formed.
```

# Successful Proof using Semantically Guarded Coinduction

```
Variable Node: Type.
Variable R: relation Node.

Inductive step (X: Node -> Prop) (x: Node) : Prop :=
| step_intro: forall y, R x y -> X y -> step X x.
Hint Constructors step.

Definition infpath := pacol step bot1.
Lemma step_mon : monotone1 step. Proof. pmonauto. Qed.
Hint Resolve step_mon : paco.

Lemma Park's_principle:
  forall (P: Node -> Prop),
  (forall x, P x -> exists y, clos_trans_in _ R x y /\ P y) ->
  forall x, P x -> infpath x.
Proof.
  pcofix CIH intros P M x Px.
  destruct (M _ Px) as (y & C & Py); clear Px.
  induction C; pfold; eauto.
Qed.
□
- \--- y.v          50% (56,0)      (Coq Script(0-) Holes)
Park's_principle is defined
```

# Successful Proof using Semantically Guarded Coinduction

```
Variable Node: Type.
Variable R: relation Node.

Inductive step (X: Node -> Prop) (x: Node) : Prop :=
| step_intro: forall y, R x y -> X y -> step X x.
Hint Constructors step.

Definition infpath := paco1 step bot1.
Lemma step_mon : monotone1 step. Proof. pmonauto. Qed.
Hint Resolve step_mon : paco.

Lemma Park's_principle:
  forall (P: Node -> Prop),
  (forall x, P x -> exists y, clos_trans_in _ R x y /\ P y) ->
  forall x, P x -> infpath x.
Proof.
  pcofix CIH intros P M x Px.
  destruct (M _ Px) as (y & C & Py); clear Px.
  induction C; pfold; eauto.
Qed.
```

```
--- y.v 50% (50,0) (Coq Script(0-) Holes)
Park's_principle is defined
```

# Successful Proof using Semantically Guarded Coinduction

Possible because of  
NO Syntactic Guardedness Checking

```
Definition infpath := paco1 step bot1.  
Lemma step_mon : monotone1 step. Proof. pmonauto. Qed.  
Hint Resolve step_mon : paco.  
  
Lemma Park's_principle:  
  forall (P: Node -> Prop),  
  (forall x, P x -> exists y, clos_trans_in _ R x y /\ P y) ->  
  forall x, P x -> infpath x.  
Proof.  
  pcofix CIH intros P M x Px.  
  destruct (M _ Px) as (y & C & Py); clear Px.  
  induction C; pfold; eauto.  
Qed.
```

```
|  
- \--- y.v          50% (50,0) (Coq Script(0-) Holes)
```

```
Park's_principle is defined
```



# Syntactic Guardedness is NOT Compositional

$$X \subseteq \nu f$$

# Syntactic Guardedness is NOT Compositional

$$X \subseteq \nu f \Rightarrow Y \subseteq \nu f$$

$$Y \subseteq \nu f \Rightarrow X \subseteq \nu f$$

---

$$X \subseteq \nu f$$

# Syntactic Guardedness is NOT Compositional

$\text{pf}_1: X \subseteq \nu f \Rightarrow Y \subseteq \nu f$  ( $\text{pf}_1$  is guarded)

$\text{pf}_2: Y \subseteq \nu f \Rightarrow X \subseteq \nu f$  ( $\text{pf}_2$  is guarded)

---

$X \subseteq \nu f$

Not Expressible in the logic

$\text{pf}_1: X \subseteq \nu f \Rightarrow Y \subseteq \nu f$  ( $\text{pf}_1$  is guarded)

$\text{pf}_2: Y \subseteq \nu f \Rightarrow X \subseteq \nu f$  ( $\text{pf}_2$  is guarded)

---

$X \subseteq \nu f$

# Make Proofs **Transparent**

$$\text{pf}_1: X \subseteq \nu f \Rightarrow Y \subseteq \nu f$$

$$\text{pf}_2: Y \subseteq \nu f \Rightarrow X \subseteq \nu f$$

---

$$X \subseteq \nu f$$

# Make Proofs **Transparent**

$$\text{pf}_1: X \subseteq \nu f \Rightarrow Y \subseteq \nu f$$

$$\text{pf}_2: Y \subseteq \nu f \Rightarrow X \subseteq \nu f$$

---

$$\text{pf}_2 \circ \text{pf}_1: X \subseteq \nu f \Rightarrow X \subseteq \nu f$$

---

$$X \subseteq \nu f$$

# Make Proofs **Transparent**

$$\text{pf}_1: X \subseteq \nu f \Rightarrow Y \subseteq \nu f$$

$$\text{pf}_2: Y \subseteq \nu f \Rightarrow X \subseteq \nu f$$

$$\frac{\text{pf}_2 \circ \text{pf}_1: X \subseteq \nu f \Rightarrow X \subseteq \nu f}{X \subseteq \nu f} \quad (\text{pf}_2 \circ \text{pf}_1 \text{ is guarded})$$

## Two Problems with **Transparent** Proofs

1. Slowdown
2. **NO** Abstraction

$$\text{pf}_1: X \subseteq \nu f \Rightarrow Y \subseteq \nu f$$

$$\text{pf}_2: Y \subseteq \nu f \Rightarrow X \subseteq \nu f$$

$$\frac{\text{pf}_2 \circ \text{pf}_1: X \subseteq \nu f \Rightarrow X \subseteq \nu f}{X \subseteq \nu f} \quad (\text{pf}_2 \circ \text{pf}_1 \text{ is guarded})$$



# Semantic Guardedness is Compositional

$$Y \subseteq G_f(X)$$

$$X \subseteq G_f(Y)$$

---

$$X \subseteq G_f(\emptyset)$$

Possible because  
Guardedness can be expressed in the Logic!

$$Y \subseteq G_f(X)$$

$$X \subseteq G_f(Y)$$

---

$$X \subseteq G_f(\emptyset)$$

# Problems with **Syntactic** Guardedness

1. Non-Compositional
2. Far from complete
3. Bad interaction with automation
4. Hard to debug
5. Slow

# ~~Problems with~~ Semantic Guardedness

1. Non-Compositional
2. Far from complete
3. Bad interaction with automation
4. Hard to debug
5. Slow

# What else is in the paper?

- Related Work
  - Glynn Winskel (1989)
  - Lawrence Moss (2001)
- Combination with Up-To Techniques
- Mechanization in Coq
  - Using Mendler-style recursion