Separating the Wheat from the Chaff:

Understanding (In-)Completeness of Proof Mechanisms for Separation Logic with Inductive Definitions

Neta Elad[1], Adithya Murali[2], Sharon Shoham[1]

1 TEL AVIV UNIVERSITY

2 WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

POPL 2026, Rennes, France

Separation Logic

Separation Logic

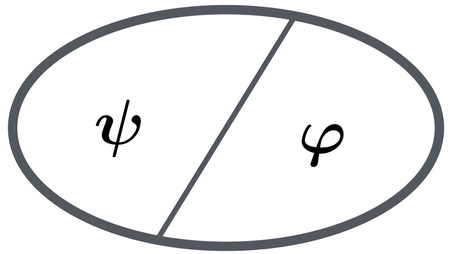
- Logic for reasoning about shared resources

Separation Logic

- Logic for reasoning about shared resources
  - We focus on memory in heap-manipulating programs

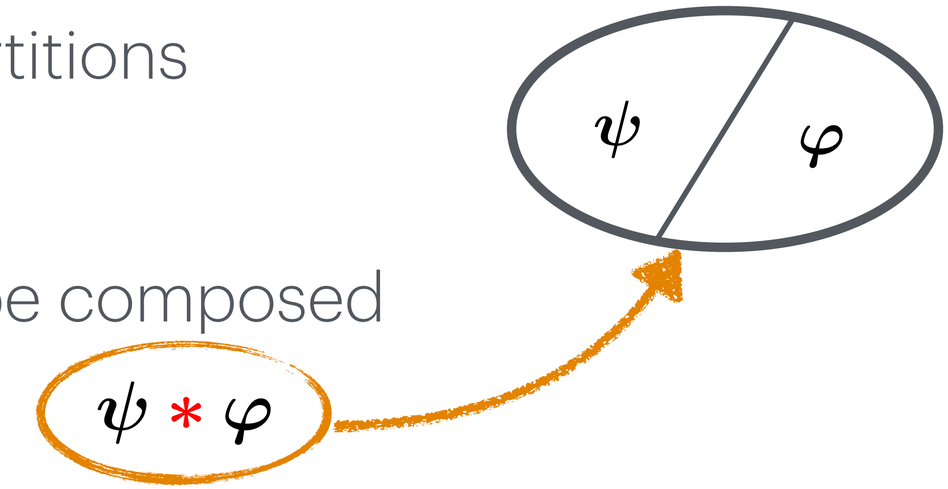
Separation Logic

- Logic for reasoning about shared resources
  - We focus on memory in heap-manipulating programs
- Formulas capture properties over partitions of the heap



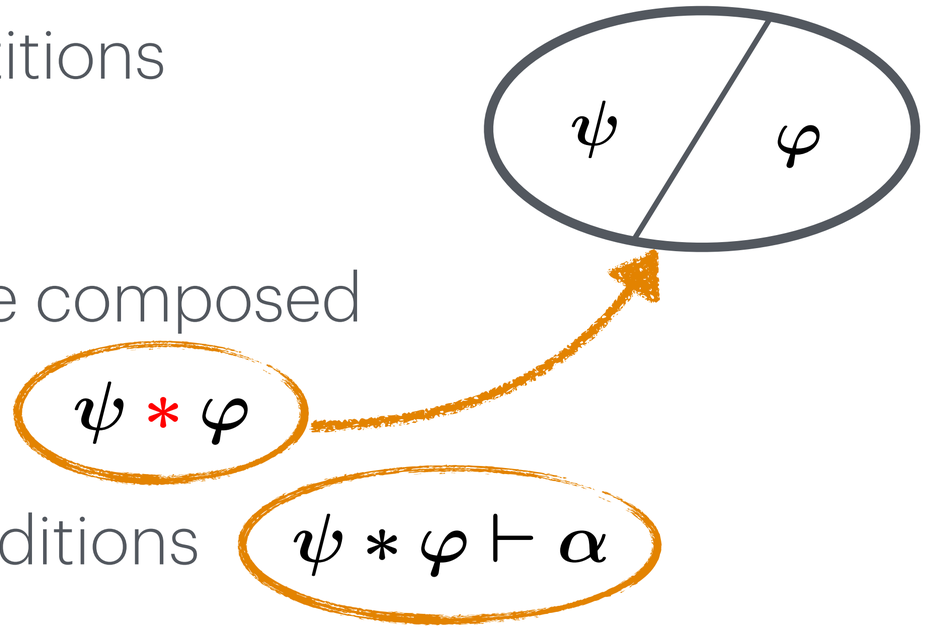
Separation Logic

- Logic for reasoning about shared resources
  - We focus on memory in heap-manipulating programs
- Formulas capture properties over partitions of the heap
  - Disjoint partitions of the heap can be composed through the separating conjunction



Separation Logic

- Logic for reasoning about shared resources
  - We focus on memory in heap-manipulating programs
- Formulas capture properties over partitions of the heap
  - Disjoint partitions of the heap can be composed through the separating conjunction
  - Entailments capture verification conditions



Separation Logic with Inductive Definitions

(SLID)

Separation Logic with Inductive Definitions

(SLID)

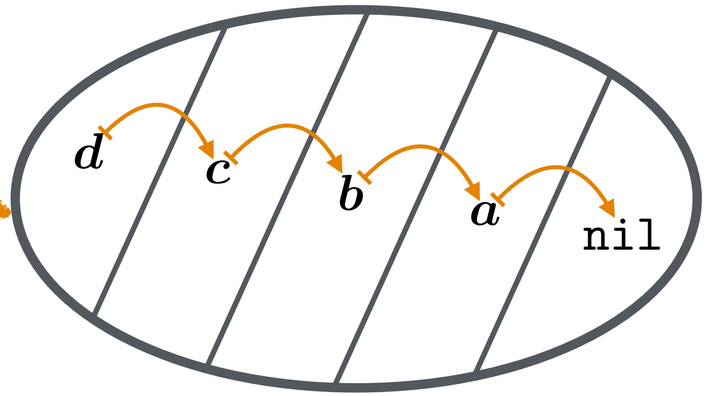
- Inductively defined relations in the heap

Separation Logic with Inductive Definitions

(SLID)

- Inductively defined relations in the heap

$$\texttt{list}(x) :\!= x = \texttt{nil} \lor \left(\exists u.x \mapsto u * \texttt{list}(u)\right)$$



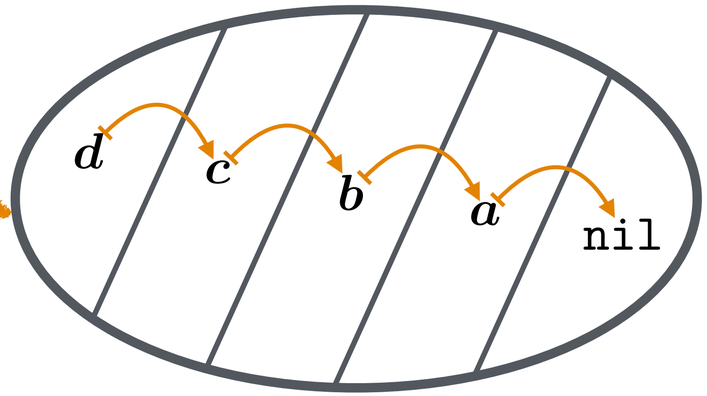
Separation Logic with Inductive Definitions

(SLID)

- Inductively defined relations in the heap

$$\texttt{list}(x) :\!= x = \texttt{nil} \lor (\exists u.x \mapsto u * \texttt{list}(u))$$

- SLID is very expressive but incomplete



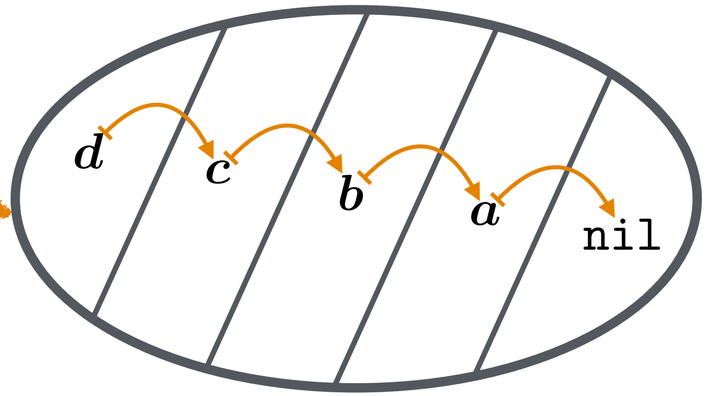
Separation Logic with Inductive Definitions

(SLID)

- Inductively defined relations in the heap

$$\texttt{list}(x) :\!= x = \texttt{nil} \lor \left(\exists u.x \mapsto u * \texttt{list}(u)\right)$$

- SLID is very expressive but incomplete



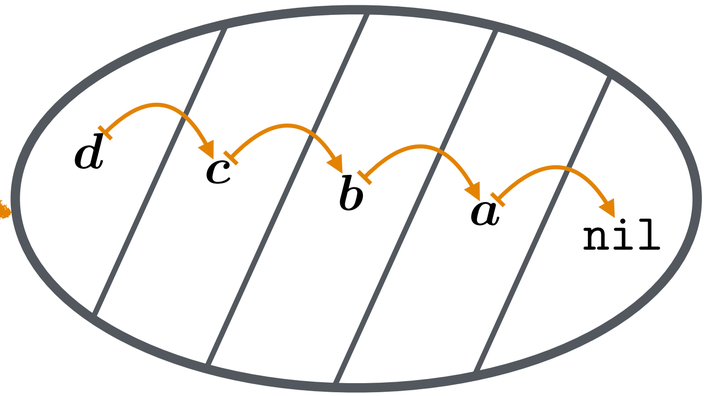
- Prominent proof mechanism: fold/unfold

Separation Logic with Inductive Definitions

(SLID)

- Inductively defined relations in the heap

$$\texttt{list}(x) :\!= x = \texttt{nil} \lor \left(\exists u.x \mapsto u * \texttt{list}(u)\right)$$



- SLID is very expressive but incomplete

- Prominent proof mechanism: fold/unfold
  - Successful in some cases but fails in others

Running Example

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \lor \left(x \neq y * \exists u. x \mapsto u * \texttt{lseg}(u, y)\right)$$

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u. x \mapsto u * \texttt{lseg}(u, y)\right)$$

- Fold/unfold proof rules

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u. x \mapsto u * \texttt{lseg}(u, y)\right)$$

- Fold/unfold proof rules
  - Fold:

$$\frac{\vdash t_1 = t_2}{\vdash \texttt{lseg}(t_1, t_2)} \qquad \frac{\vdash t_1 \neq t_2 * t_1 \mapsto t_3 * \texttt{lseg}(t_3, t_2)}{\vdash \texttt{lseg}(t_1, t_2)}$$

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

- Fold/unfold proof rules
  - Fold:

$$\frac{\vdash t_1 = t_2}{\vdash \texttt{lseg}(t_1, t_2)} \qquad \frac{\vdash t_1 \neq t_2 * t_1 \mapsto t_3 * \texttt{lseg}(t_3, t_2)}{\vdash \texttt{lseg}(t_1, t_2)}$$

  - Unfold:

$$\frac{\vdash \texttt{lseg}(t_1, t_2)}{\vdash t_1 = t_2 \vee \left(t_1 \neq t_2 * t_1 \mapsto c^\star * \texttt{lseg}(c^\star, t_2)\right)}$$

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

- Fold/unfold proof rules
  - Fold:

$$\frac{\vdash t_1 = t_2}{\vdash \texttt{lseg}(t_1, t_2)} \qquad \frac{\vdash t_1 \neq t_2 * t_1 \mapsto t_3 * \texttt{lseg}(t_3, t_2)}{\vdash \texttt{lseg}(t_1, t_2)}$$

  - Unfold:

Fresh constant

$$\frac{\vdash \texttt{lseg}(t_1, t_2)}{\vdash t_1 = t_2 \vee \left(t_1 \neq t_2 * t_1 \mapsto c^\star * \texttt{lseg}(c^\star, t_2)\right)}$$

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \lor \left(x \neq y * \exists u. x \mapsto u * \texttt{lseg}(u, y)\right)$$

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \lor \left(x \neq y * \exists u. x \mapsto u * \texttt{lseg}(u, y)\right)$$

$$a \neq c * a \mapsto b * \texttt{lseg}(b, c) \vdash \texttt{lseg}(a, c)$$

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

A list-segment can extended from the *left*

$$a \neq c * a \mapsto b * \texttt{lseg}(b, c) \vdash \texttt{lseg}(a, c)$$

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

A list-segment can extended from the *left*

$$a \neq c * a \mapsto b * \texttt{lseg}(b, c) \vdash \texttt{lseg}(a, c)$$

Provable by a single fold

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \lor \left(x \neq y * \exists u. x \mapsto u * \texttt{lseg}(u, y)\right)$$

A list-segment can extended from the *left*

$$a \neq c * a \mapsto b * \texttt{lseg}(b, c) \vdash \texttt{lseg}(a, c)$$

Provable by a single fold

$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \lor \left(x \neq y * \exists u. x \mapsto u * \texttt{lseg}(u, y)\right)$$

A list-segment can extended from the *left*

$$a \neq c * a \mapsto b * \texttt{lseg}(b, c) \vdash \texttt{lseg}(a, c)$$

Provable by a single fold

A list-segment can extended from the *right*

$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

A list-segment can extended from the *left*

$$a \neq c * a \mapsto b * \texttt{lseg}(b, c) \vdash \texttt{lseg}(a, c)$$

Provable by a single fold

A list-segment can extended from the *right*

$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$

Not provable by any fold/unfold

Running Example

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

A list-segment can extended from the *left*

$$a \neq c * a \mapsto b * \texttt{lseg}(b, c) \vdash \texttt{lseg}(a, c)$$

Provable by a single fold

A list-segment can extended from the *right*

$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$

Not provable by any fold/unfold

Why? And more generally, when?

Understanding Fold/Unfold Failures

Understanding Fold/Unfold Failures

- Why fails? Because fold/unfold captures weaker semantics than SLID

Understanding Fold/Unfold Failures

- Why fails? Because fold/unfold captures weaker semantics than SLID

- When fails? When *rogue* counter-models exist

Understanding Fold/Unfold Failures

- Why fails? Because fold/unfold captures weaker semantics than SLID

- When fails? When *rogue* counter-models exist
  - Rogue meaning models not considered by SLID semantics

Outline

Outline

- Weak Separation Logic (WSL): weaker semantics of separation logic

Outline

- Weak Separation Logic (WSL): weaker semantics of separation logic
  - Fold/unfold is *sound* for WSL

Outline

- Weak Separation Logic (WSL): weaker semantics of separation logic
  - Fold/unfold is *sound* for WSL
- Rogue models of WSL

Outline

- Weak Separation Logic (WSL): weaker semantics of separation logic
  - Fold/unfold is *sound* for WSL
- Rogue models of WSL
  - How to find them

Outline

- Weak Separation Logic (WSL): weaker semantics of separation logic
  - Fold/unfold is *sound* for WSL
- Rogue models of WSL
  - How to find them
  - Do they capture all fold/unfold failures? Is fold/unfold *complete* for WSL?

Outline

- Weak Separation Logic (WSL): weaker semantics of separation logic
  - Fold/unfold is *sound* for WSL
- Rogue models of WSL
  - How to find them
  - Do they capture all fold/unfold failures? Is fold/unfold *complete* for WSL?
- Rogue models in practice

Outline

- Weak Separation Logic (WSL): weaker semantics of separation logic
  - Fold/unfold is *sound* for WSL
- Rogue models of WSL
  - How to find them
  - Do they capture all fold/unfold failures? Is fold/unfold *complete* for WSL?
- Rogue models in practice
- Conclusion

Weak Separation Logic (WSL)

Weak Separation Logic (WSL)

- WSL relaxes key requirements of SLID that are sources of incompleteness:

Weak Separation Logic (WSL)

- WSL relaxes key requirements of SLID that are sources of incompleteness:
  - Heaps in WSL are allowed to be **infinite**

Weak Separation Logic (WSL)

- WSL relaxes key requirements of SLID that are sources of incompleteness:
  - Heaps in WSL are allowed to be **infinite**
  - Interpretations of inductive definitions are allowed to be **any fixpoint**

Weak Separation Logic (WSL)

- WSL relaxes key requirements of SLID that are sources of incompleteness:
  - Heaps in WSL are allowed to be **infinite**
  - Interpretations of inductive definitions are allowed to be **any fixpoint**
  - (*) Also relaxes semantics of background theories, see paper

Weak Separation Logic (WSL)

- WSL relaxes key requirements of SLID that are sources of incompleteness:
  - Heaps in WSL are allowed to be **infinite**
  - Interpretations of inductive definitions are allowed to be **any fixpoint**
  - (*) Also relaxes semantics of background theories, see paper
- WSL contains strictly more models than SLID

Weak Separation Logic (WSL)

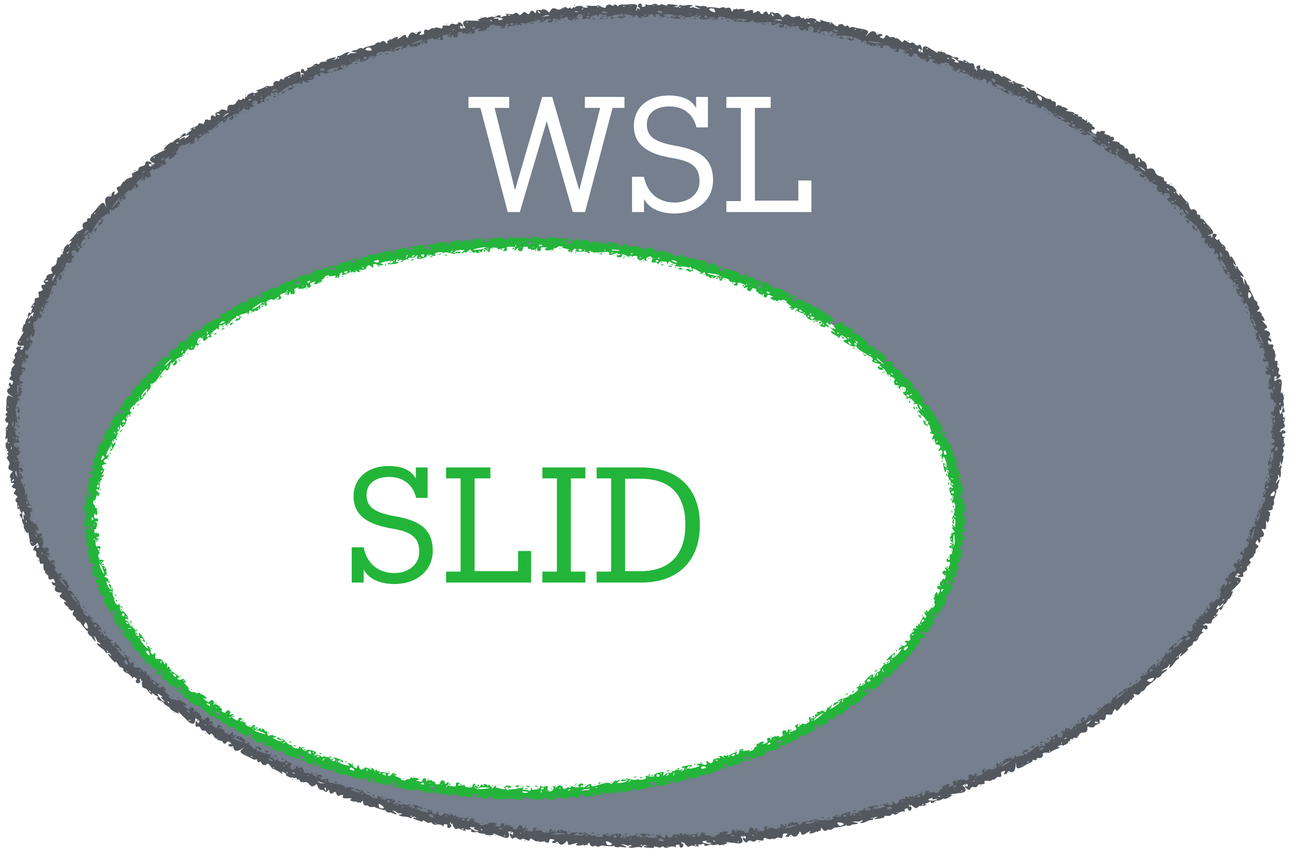
- WSL relaxes key requirements of SLID that are sources of incompleteness:
  - Heaps in WSL are allowed to be **infinite**
  - Interpretations of inductive definitions are allowed to be **any fixpoint**
  - (*) Also relaxes semantics of background theories, see paper
- WSL contains strictly more models than SLID

Theorem 5.4. Fold/unfold is sound for WSL

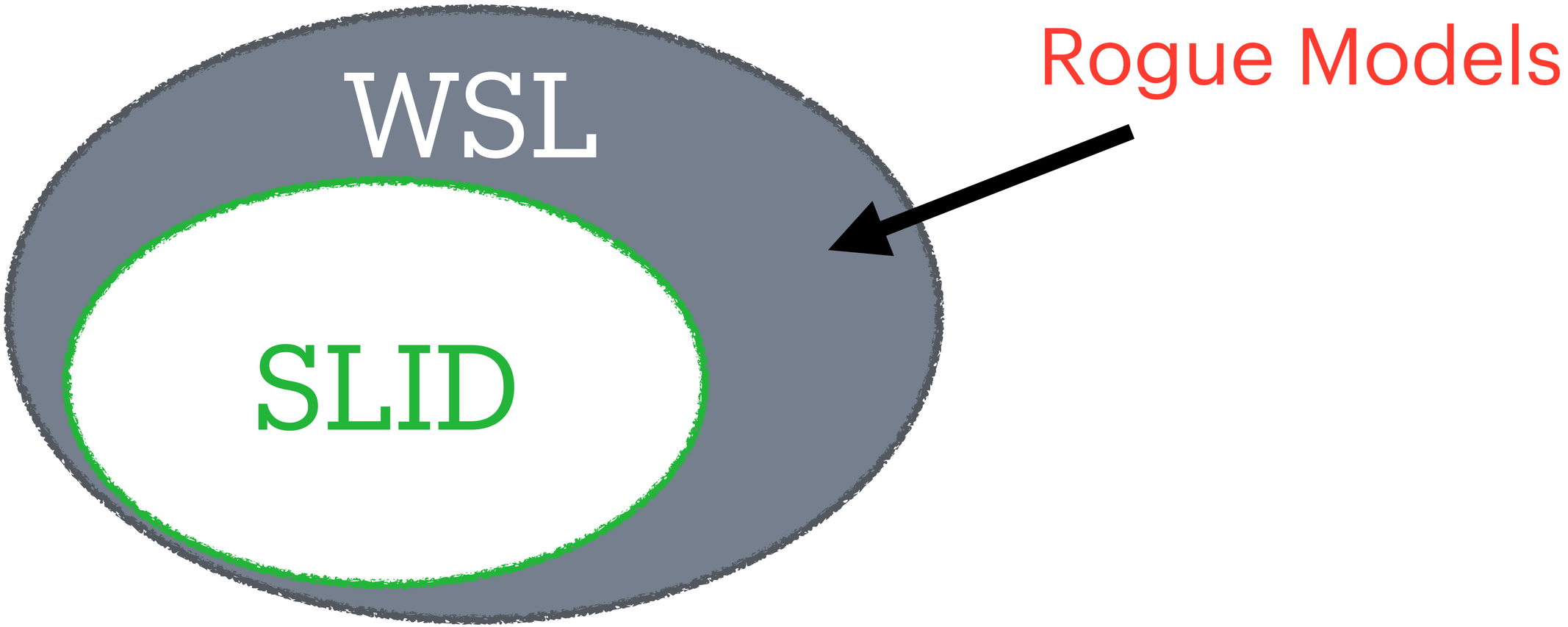
Rogue Models of WSL

Rogue Models of WSL

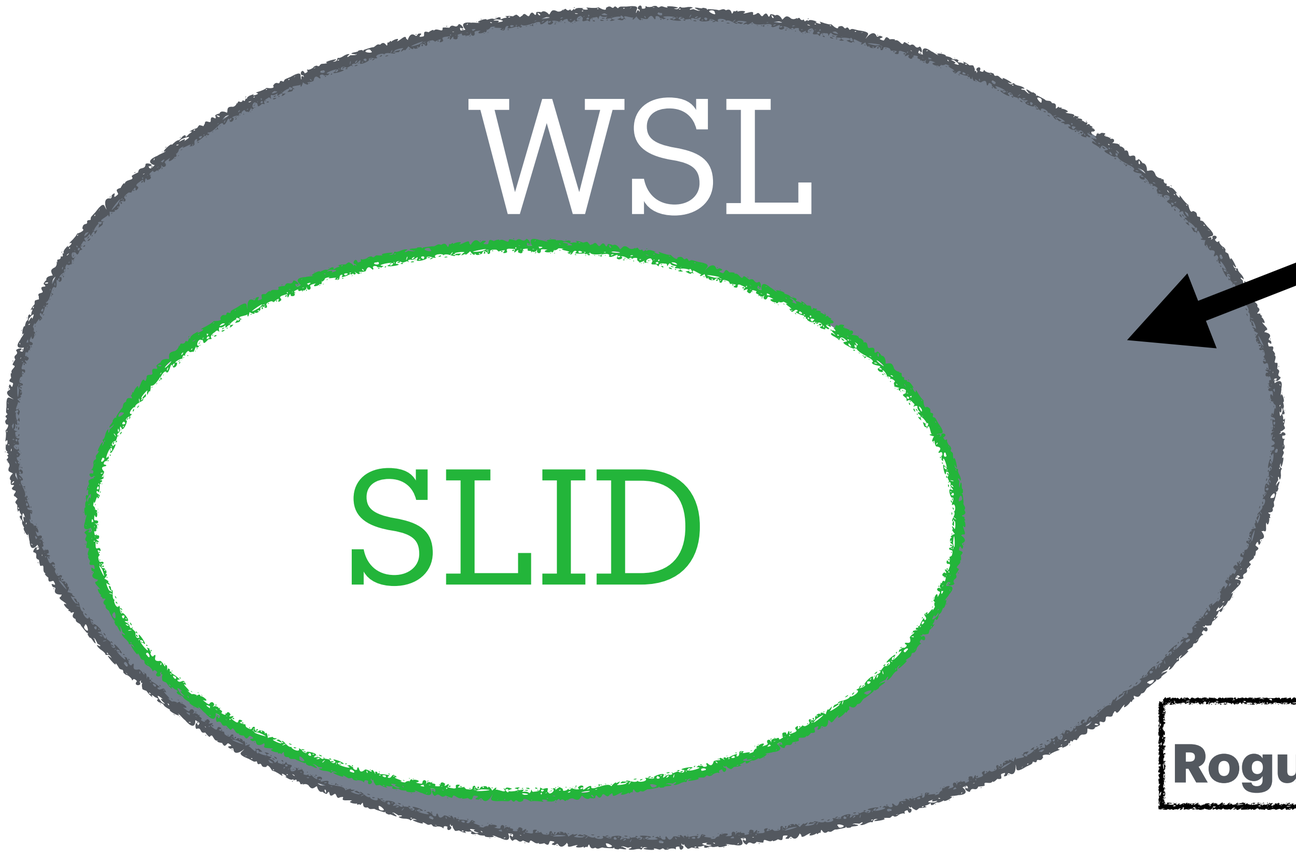
Rogue Models of WSL



Rogue Models of WSL



Rogue Models of WSL



Rogue Models

Rogue counter-models ⇒ fold/unfold failures

Rogue Counter-Model

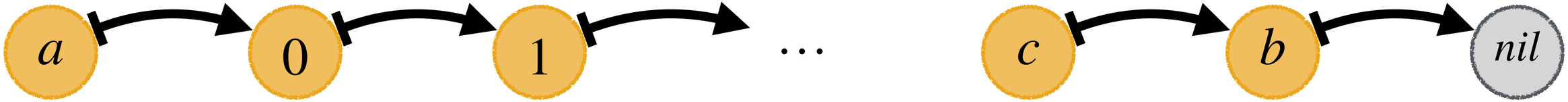
$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u. x \mapsto u * \texttt{lseg}(u, y)\right)$$

$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$

Rogue Counter-Model

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

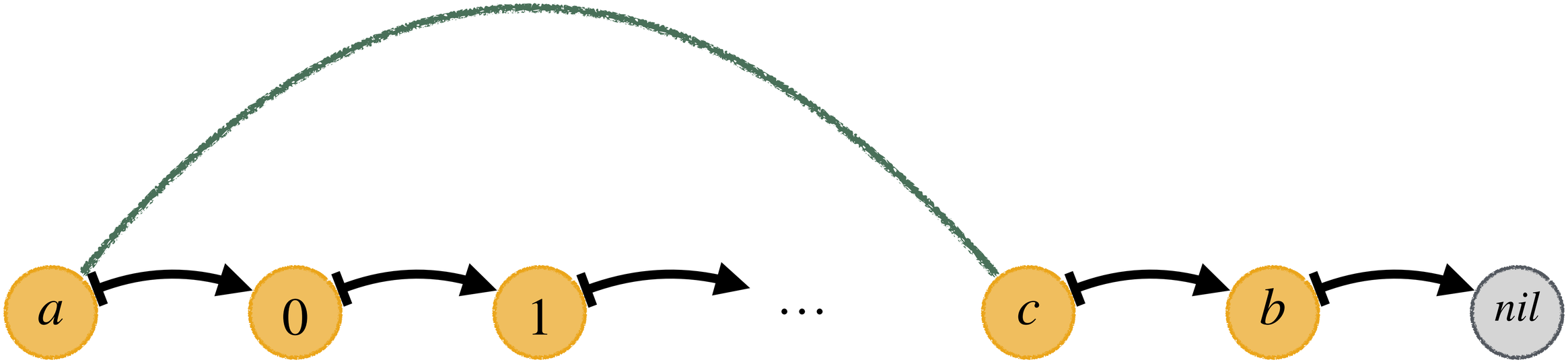
$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$



Rogue Counter-Model

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u. x \mapsto u * \texttt{lseg}(u, y)\right)$$

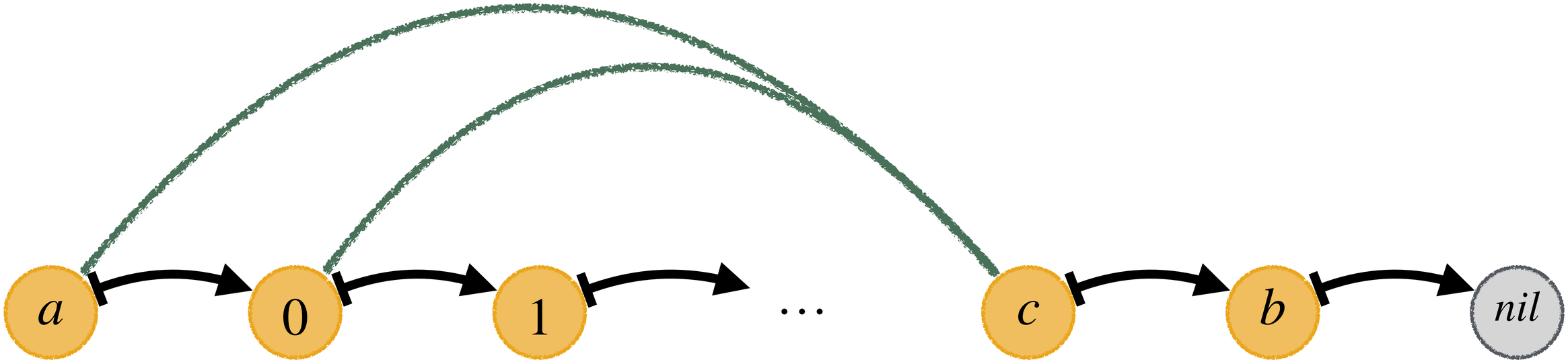
$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$



Rogue Counter-Model

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

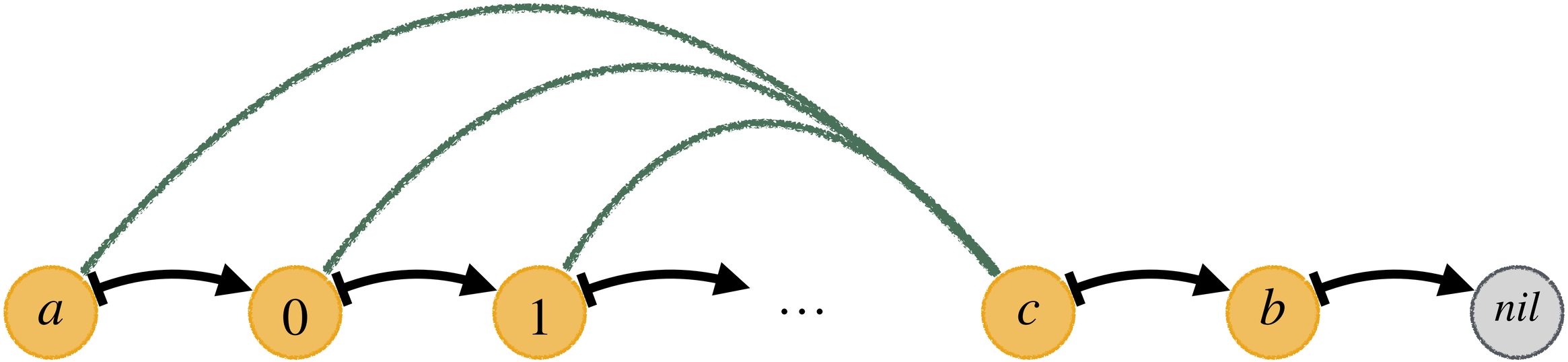
$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$



Rogue Counter-Model

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

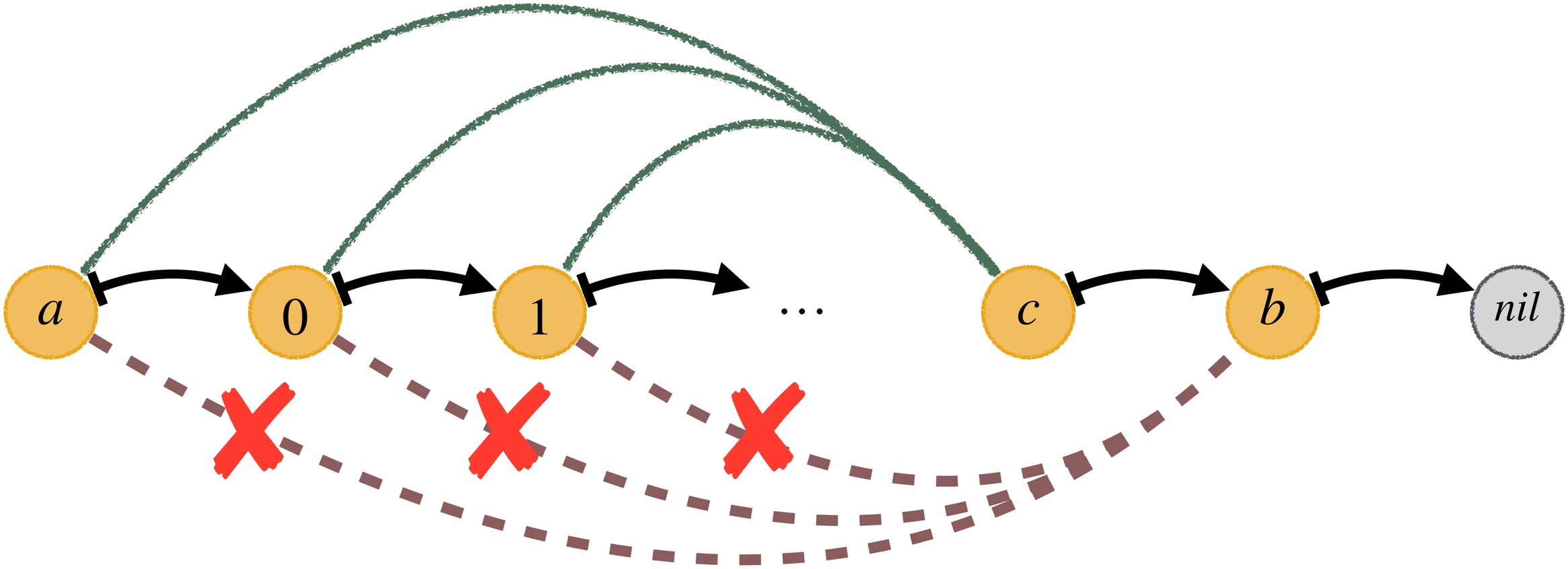
$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$



Rogue Counter-Model

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lseg}(a, b) * b \mapsto \texttt{nil}$$

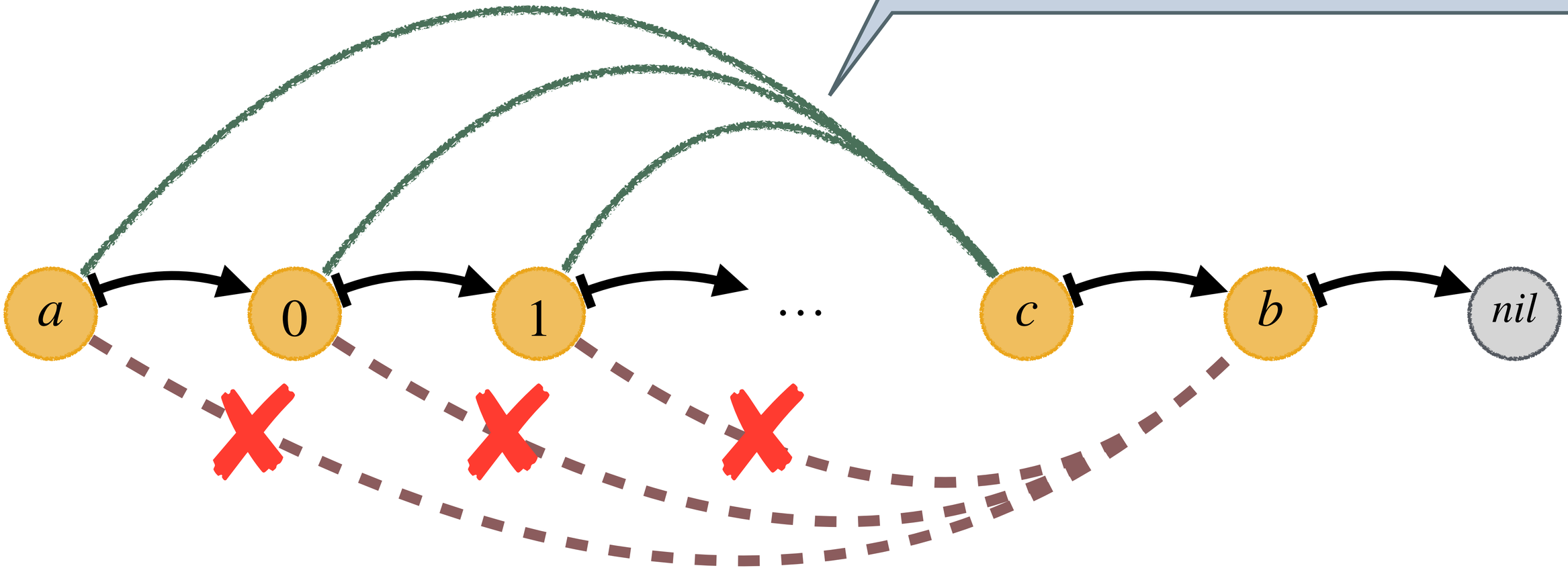


Rogue Counter-Model

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{l}$$

An actionable explanation of proof failure

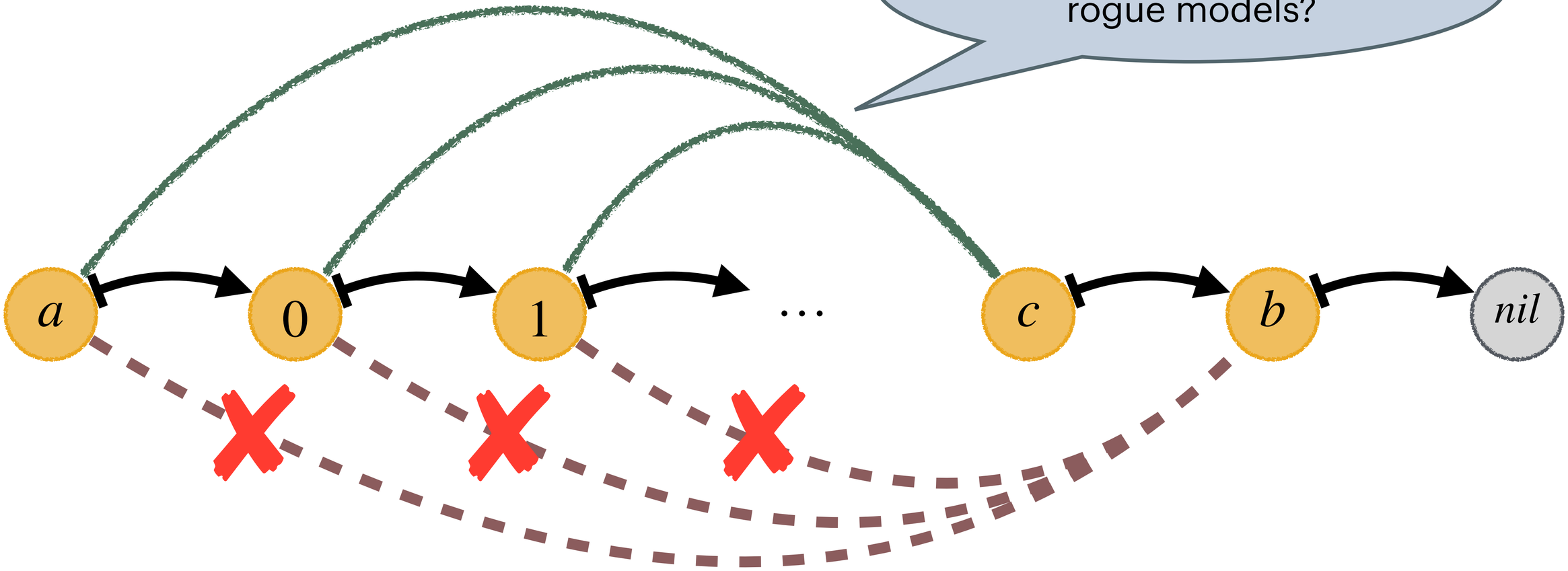


Rogue Counter-Model

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lse}$$

How can we find these rogue models?



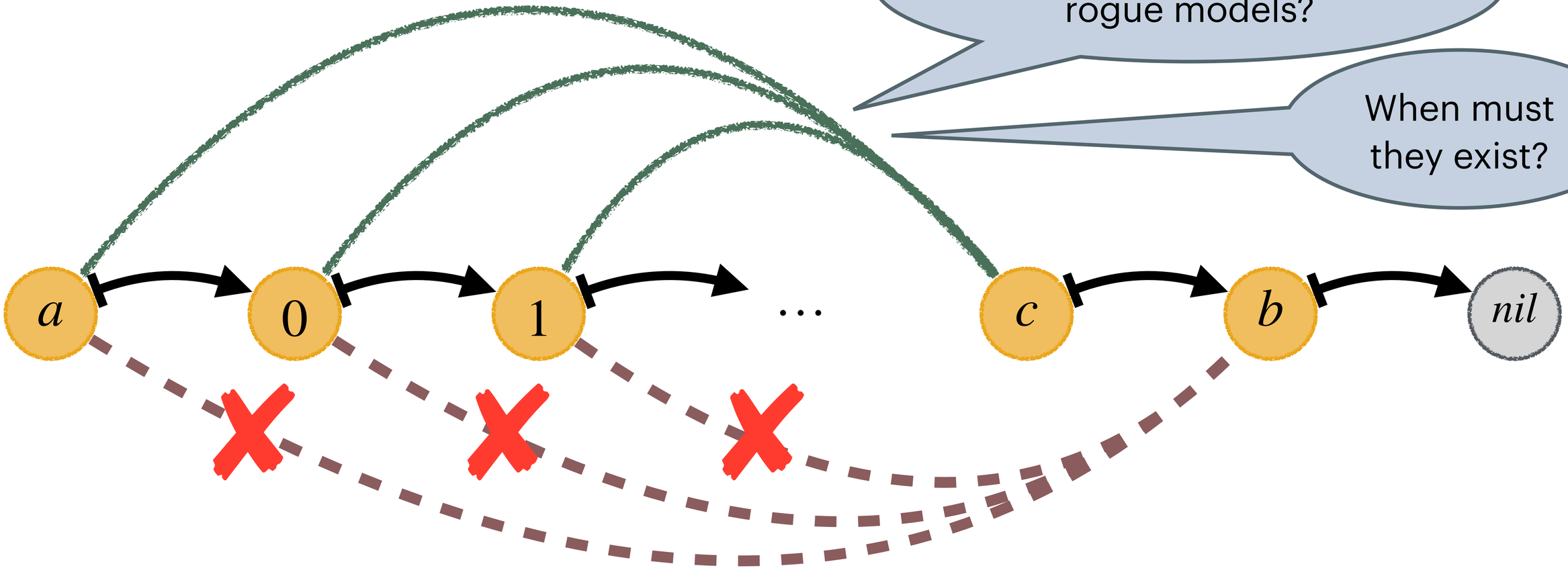
Rogue Counter-Model

$$\texttt{lseg}(x, y) \coloneqq x = y \vee \left(x \neq y * \exists u.x \mapsto u * \texttt{lseg}(u, y)\right)$$

$$\texttt{lseg}(a, c) * c \mapsto b * b \mapsto \texttt{nil} \vdash \texttt{lse}$$

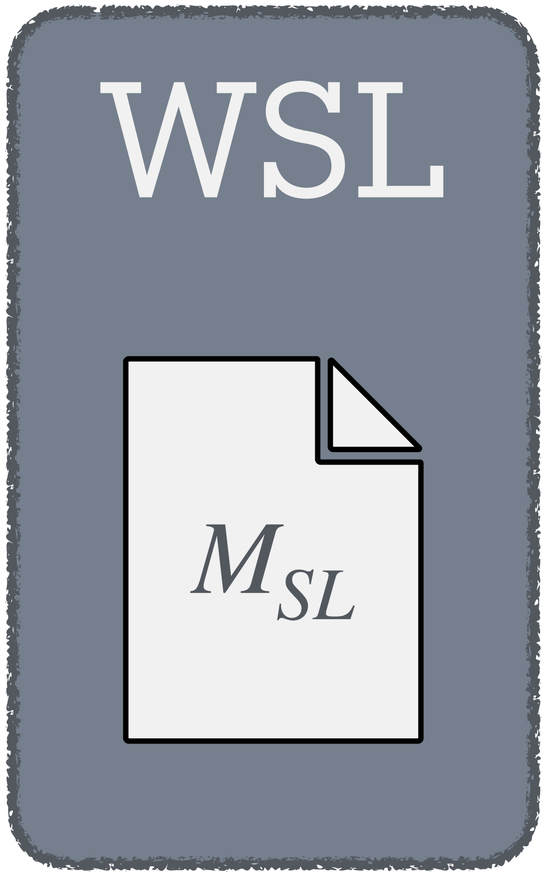
How can we find these rogue models?

When must they exist?

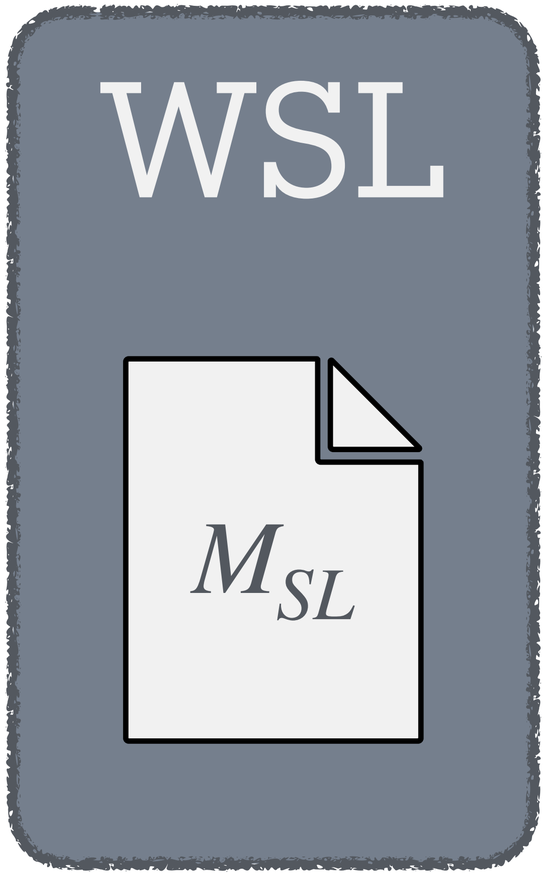


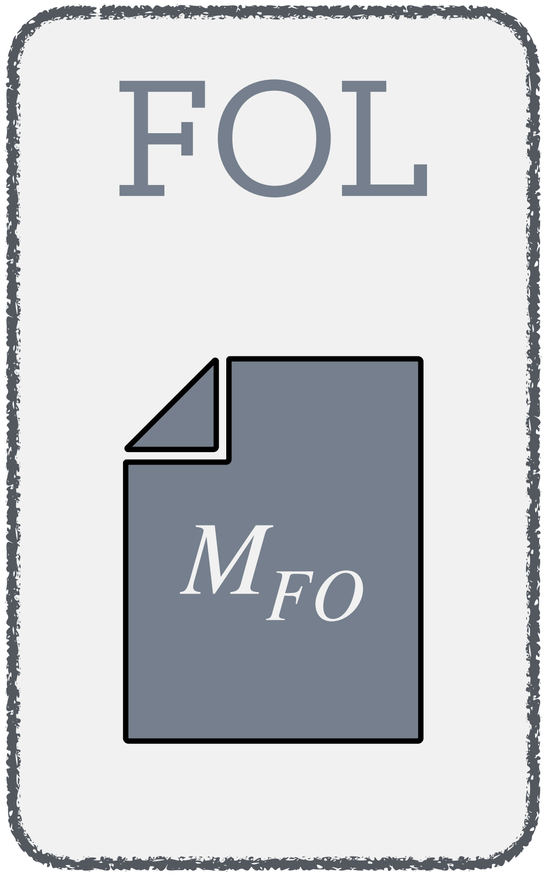
Finding Rogue Models

Finding Rogue Models

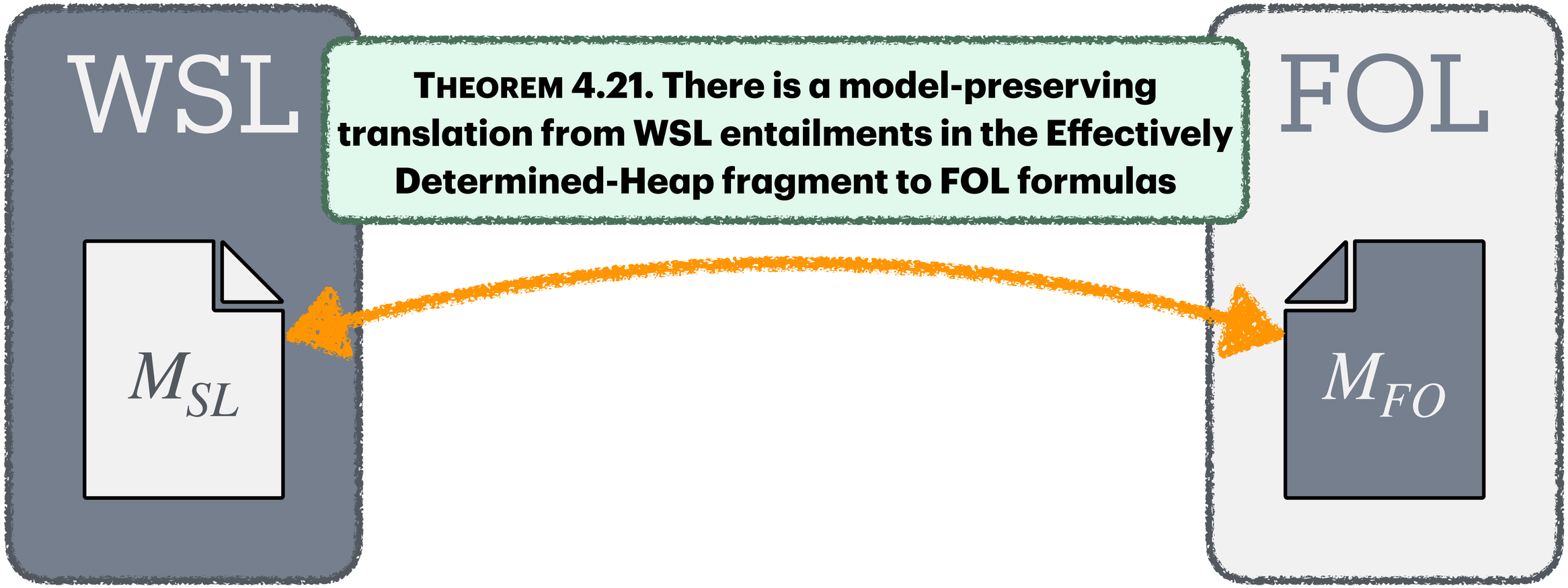


Finding Rogue Models





Finding Rogue Models



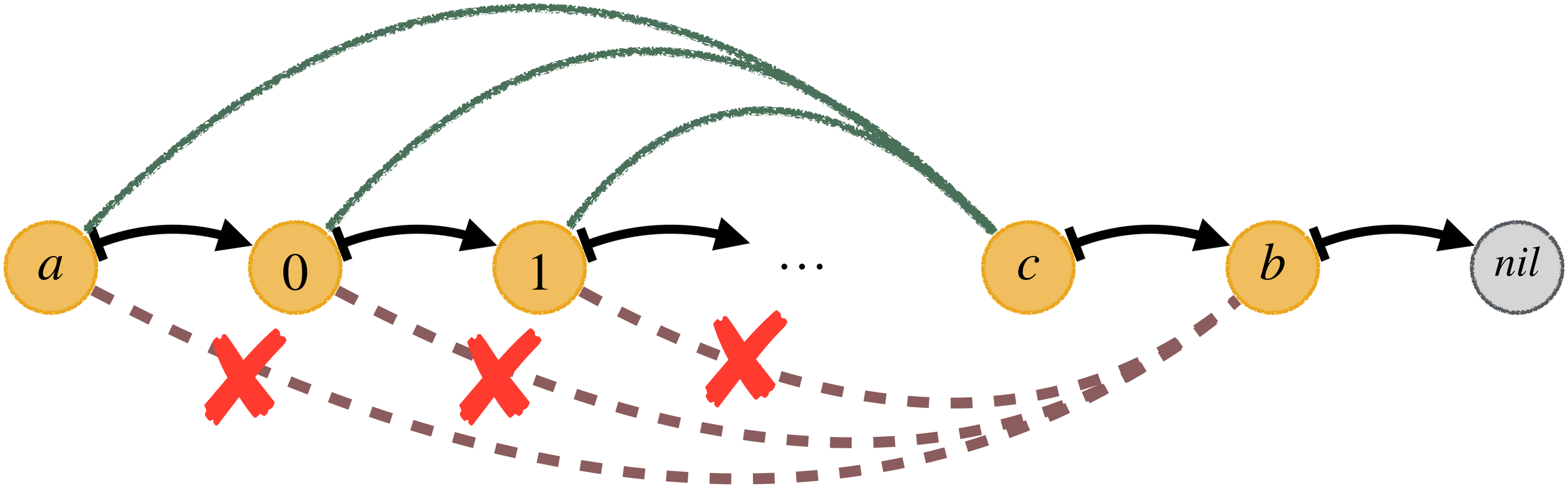
Rogue Models as Symbolic Structures

- On the FOL side we can use the formalism of *symbolic structures*[1] to represent and find these rogue models:

[1] “An Infinite Needle in a Finite Haystack: Finding Infinite Counter-Models in Deductive Verification”, **Elad** et al.

Rogue Models as Symbolic Structures

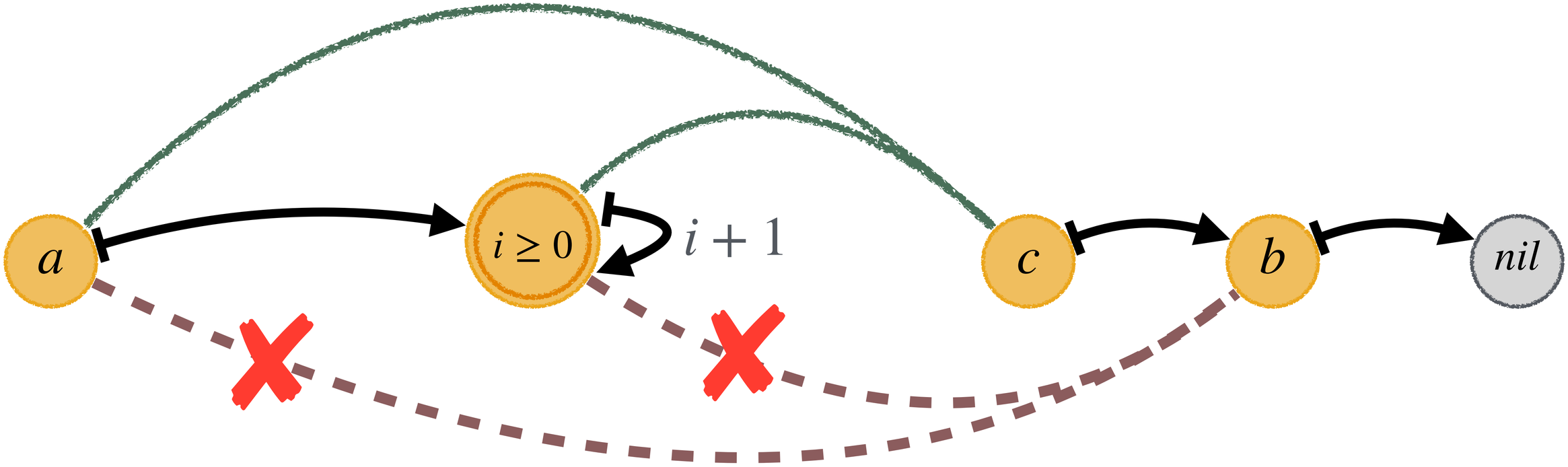
- On the FOL side we can use the formalism of *symbolic structures*[1] to represent and find these rogue models:



[1] “An Infinite Needle in a Finite Haystack: Finding Infinite Counter-Models in Deductive Verification”, **Elad** et al.

Rogue Models as Symbolic Structures

- On the FOL side we can use the formalism of *symbolic structures*[1] to represent and find these rogue models:



[1] "An Infinite Needle in a Finite Haystack: Finding Infinite Counter-Models in Deductive Verification", **Elad** et al.

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failu

Inductive definitions still contain quantifiers

Theorem 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

Entailment with inductive definitions is valid in WSL

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

Entailment with inductive definitions is valid in WSL

THEOREM 4.21

Set of quantified formulas is unsatisfiable in FOL

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

Entailment with inductive definitions is valid in WSL

THEOREM 4.21

Set of quantified formulas is unsatisfiable in FOL

HERBRAND'S THEOREM

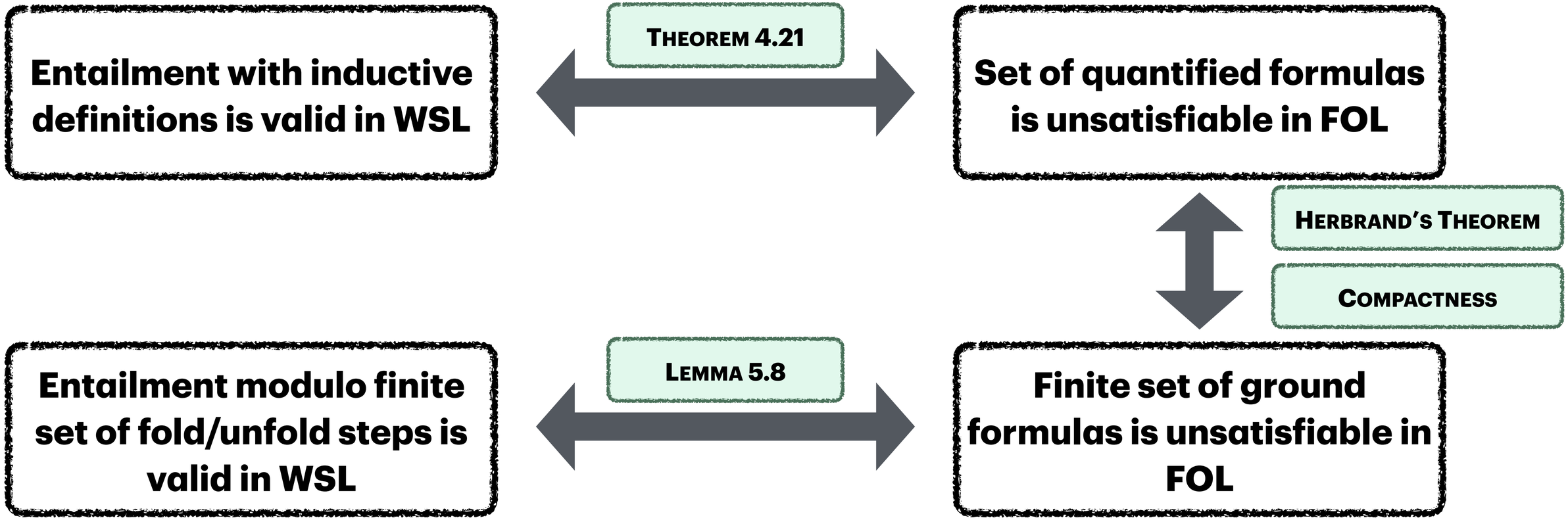
COMPACTNESS

Finite set of ground formulas is unsatisfiable in FOL

Rogue Models of WSL and Fold/Unfold Failures

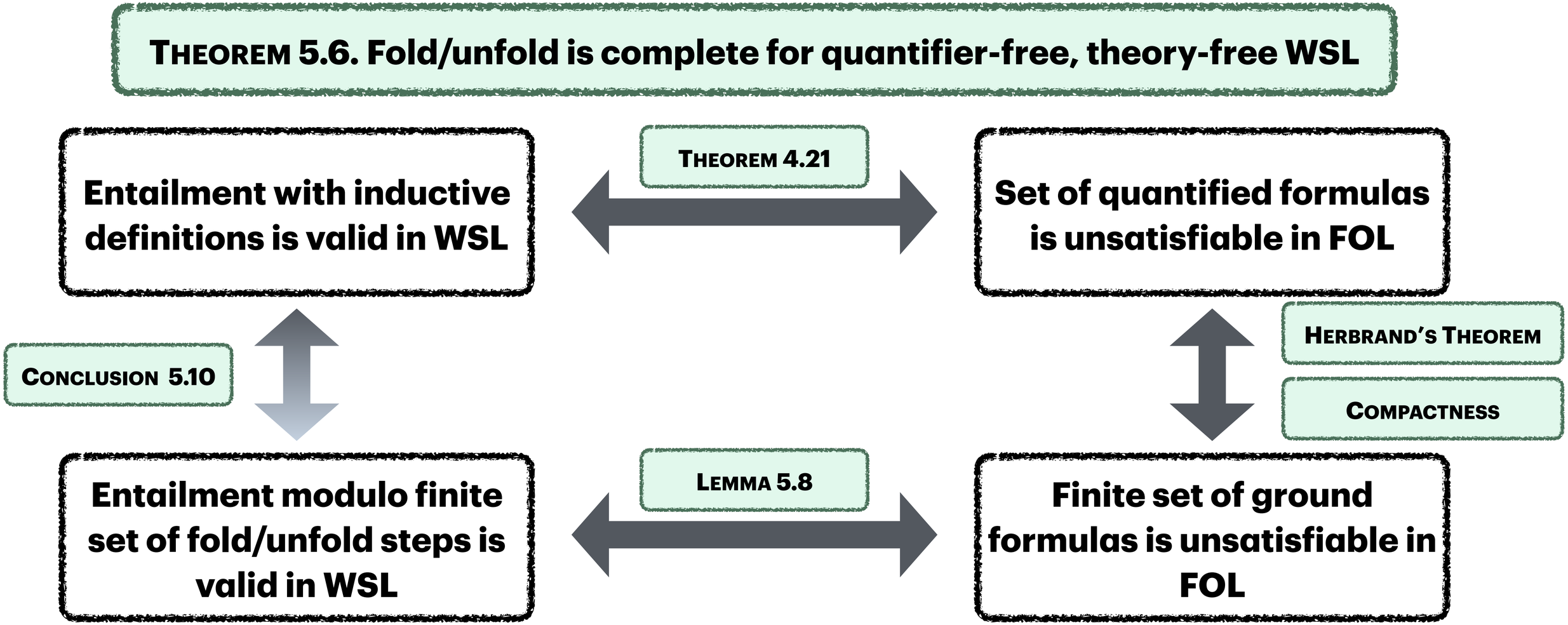
Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL



Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?



Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

- All proof failures are due to the existence of rogue WSL counter-models

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

- All proof failures are due to the existence of rogue WSL counter-models

Fold/unfold failures ⇔ WSL rogue counter-models

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

- All proof failures are due to the existence of rogue WSL counter-models

Fold/unfold failures ⇔ WSL rogue counter-models

- The “best” fragment we could hope for

Rogue Models of WSL and Fold/Unfold Failures

Do rogue models of WSL capture all fold/unfold failures?

THEOREM 5.6. Fold/unfold is complete for quantifier-free, theory-free WSL

- All proof failures are due to the existence of rogue WSL counter-models

Fold/unfold failures ⇔ WSL rogue counter-models

- The “best” fragment we could hope for

Fold/unfold alone has no way to reason about theories or do any quantifier instantiation beyond the instantiation of inductive definitions

Completeness of WSL Beyond Fold/Unfold

Completeness of WSL Beyond Fold/Unfold

- The translation to FOL works for the entire EDH fragment

Completeness of WSL Beyond Fold/Unfold

- The translation to FOL works for the entire EDH fragment

- This gives a complete proof mechanism for EDH: translating to FOL and using a complete proof system of FOL

Completeness of WSL Beyond Fold/Unfold

- The translation to FOL works for the entire EDH fragment

- This gives a complete proof mechanism for EDH: translating to FOL and using a complete proof system of FOL

THEOREM 4.22. The Effectively Determined-Heap fragment of WSL admits a complete proof mechanism

Evaluation

Evaluation

- Prototype implementation, artifact available

Evaluation

- Prototype implementation, artifact available
  - Implements the WSL→FOL translation

Evaluation

- Prototype implementation, artifact available
  - Implements the WSL→FOL translation
  - Extends FEST[1] to find rogue models

[1] “An Infinite Needle in a Finite Haystack: Finding Infinite Counter-Models in Deductive Verification”, **Elad** et al.

Evaluation

- Prototype implementation, artifact available
  - Implements the WSL→FOL translation
  - Extends FEST[1] to find rogue models
- SLS[2] benchmark with >700 examples, divided into 5 parts

[1] “An Infinite Needle in a Finite Haystack: Finding Infinite Counter-Models in Deductive Verification”, **Elad** et al.

[2] “Automated lemma synthesis in symbolic-heap separation logic”, Ta et al.

Evaluation

- Prototype implementation, artifact available
  - Implements the WSL→FOL translation
  - Extends FEST[1] to find rogue models
- SLS[2] benchmark with >700 examples, divided into 5 parts
  - Found rogue models for 92% examples in one of the parts

[1] “An Infinite Needle in a Finite Haystack: Finding Infinite Counter-Models in Deductive Verification”, **Elad** et al.

[2] “Automated lemma synthesis in symbolic-heap separation logic”, Ta et al.

Conclusion

Conclusion

- Fold/unfold does not capture the precise semantics of SLID

Conclusion

- Fold/unfold does not capture the precise semantics of SLID
- Weak Separation Logic: weaker semantics for SLID

Conclusion

- Fold/unfold does not capture the precise semantics of SLID
- Weak Separation Logic: weaker semantics for SLID
- Fold/unfold is sound & complete for a fragment of WSL

Theorem 5.4 **Theorem 5.6**

Conclusion

- Fold/unfold does not capture the precise semantics of SLID
- Weak Separation Logic: weaker semantics for SLID
- Fold/unfold is sound & complete for a fragment of WSL
- The larger EDH fragment of WSL is complete (not through fold/unfold)

Theorem 5.4

Theorem 5.6

Theorem 4.22

Conclusion

- Fold/unfold does not capture the precise semantics of SLID
- Weak Separation Logic: weaker semantics for SLID
- Fold/unfold is sound & complete for a fragment of WSL — **Theorem 5.4** **Theorem 5.6**
- The larger EDH fragment of WSL is complete (not through fold/unfold) — **Theorem 4.22**
- The heart of the above contributions is a model-preserving translation to FOL — **Theorem 4.21**

Conclusion

- Fold/unfold does not capture the precise semantics of SLID
- Weak Separation Logic: weaker semantics for SLID
- Fold/unfold is sound & complete for a fragment of WSL **Theorem 5.4** **Theorem 5.6**
- The larger EDH fragment of WSL is complete (not through fold/unfold) **Theorem 4.22**
- The heart of the above contributions is a model-preserving translation to FOL **Theorem 4.21**
- Translation used in practice to automatically find entailment-violating models of WSL

Conclusion

- Fold/unfold does not capture the precise semantics of SLID
- Weak Separation Logic: weaker semantics for SLID
- Fold/unfold is sound & complete for a fragment of WSL — **Theorem 5.4** **Theorem 5.6**
- The larger EDH fragment of WSL is complete (not through fold/unfold) — **Theorem 4.22**
- The heart of the above contributions is a model-preserving translation to FOL — **Theorem 4.21**
- Translation used in practice to automatically find entailment-violating models of WSL

Thanks!