# Formal Foundations for Translational Separation Logic Verifiers

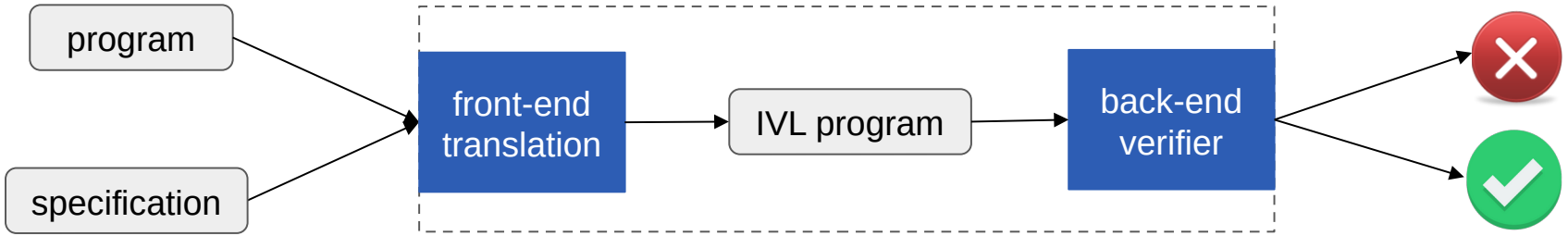
**Thibault Dardinier**[1], Michael Sammler[1], Gaurav Parthasarathy[1], Alexander J. Summers[2], and Peter Müller[1]

[1] ETH zürich

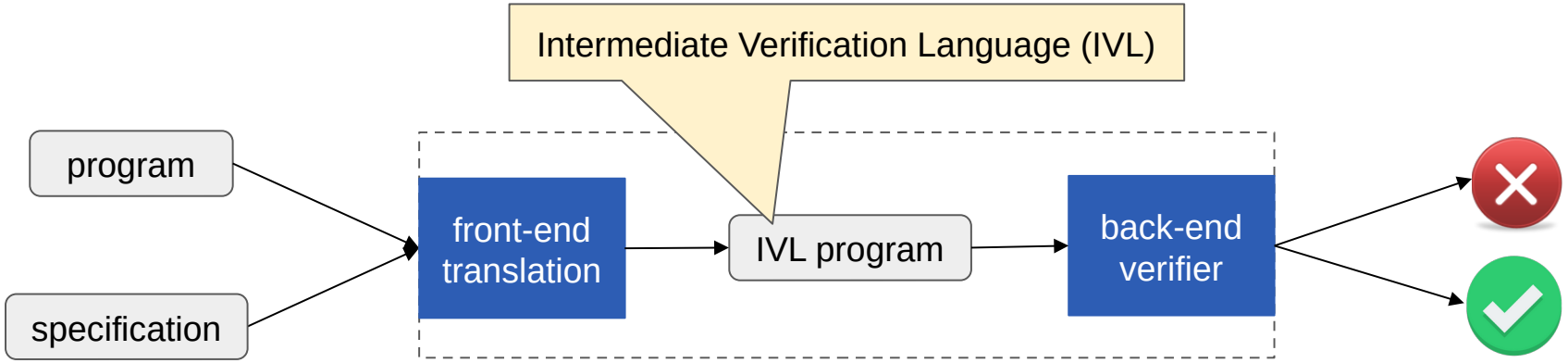
[2] UBC THE UNIVERSITY OF BRITISH COLUMBIA

# Translational Verifiers

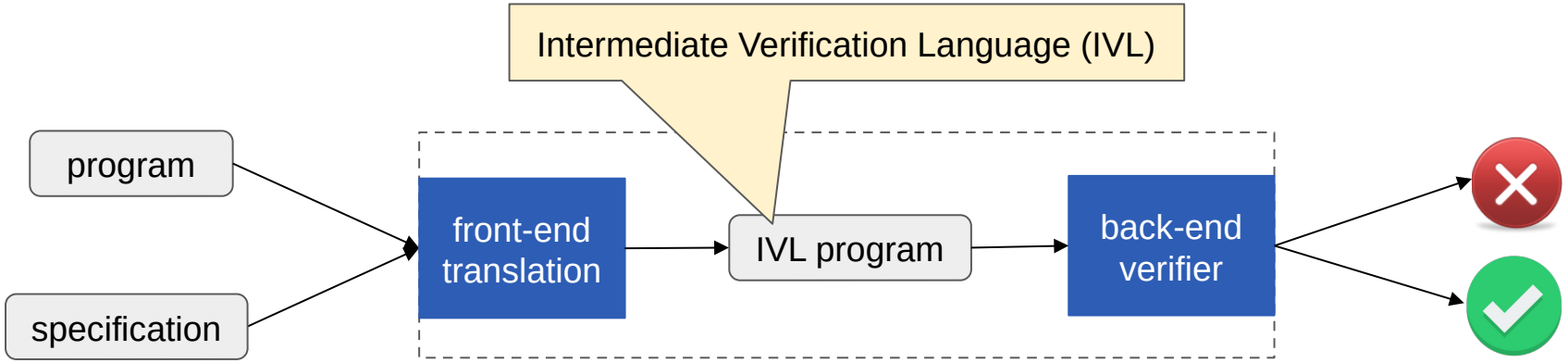
# Translational Verifiers



# Translational Verifiers



# Translational Verifiers



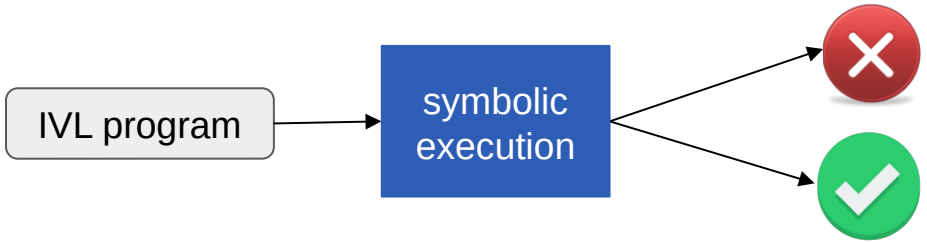

Boogie

Gillian

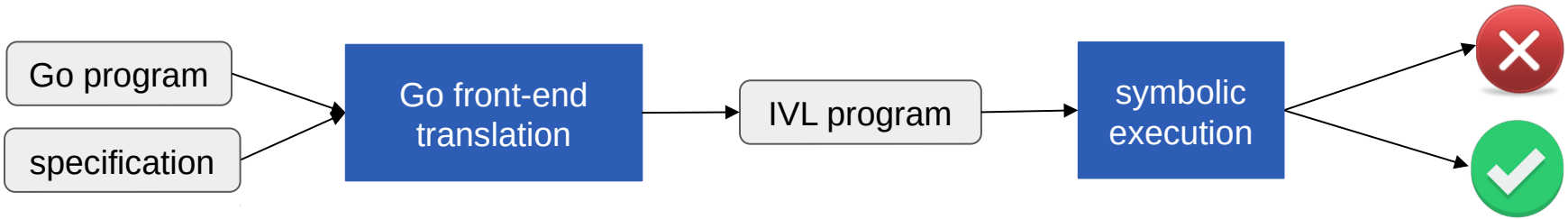
VIPER

Why3

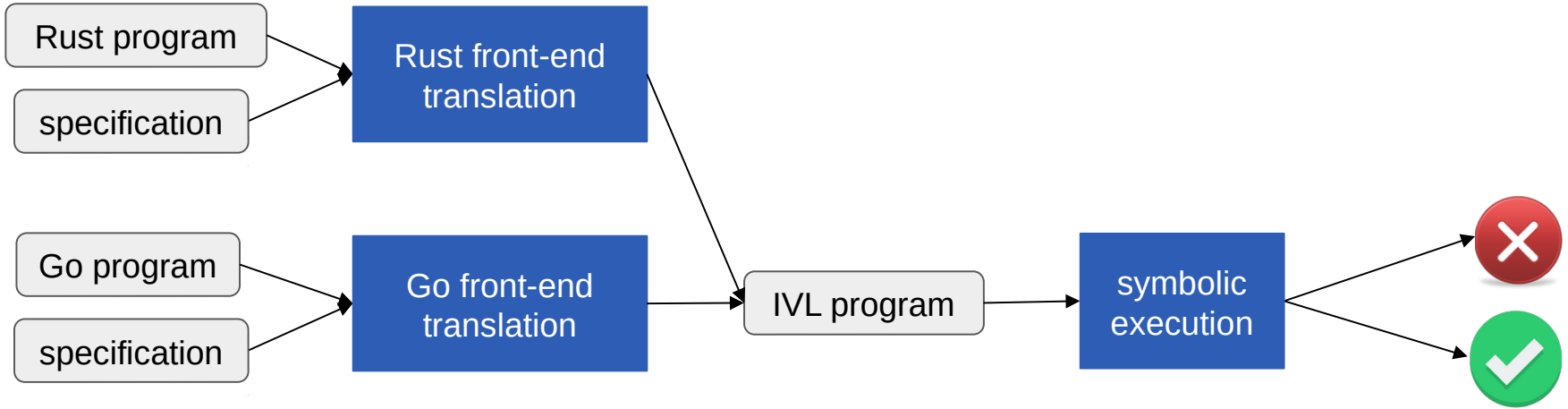
# Translational Separation Logic (SL) Verifiers



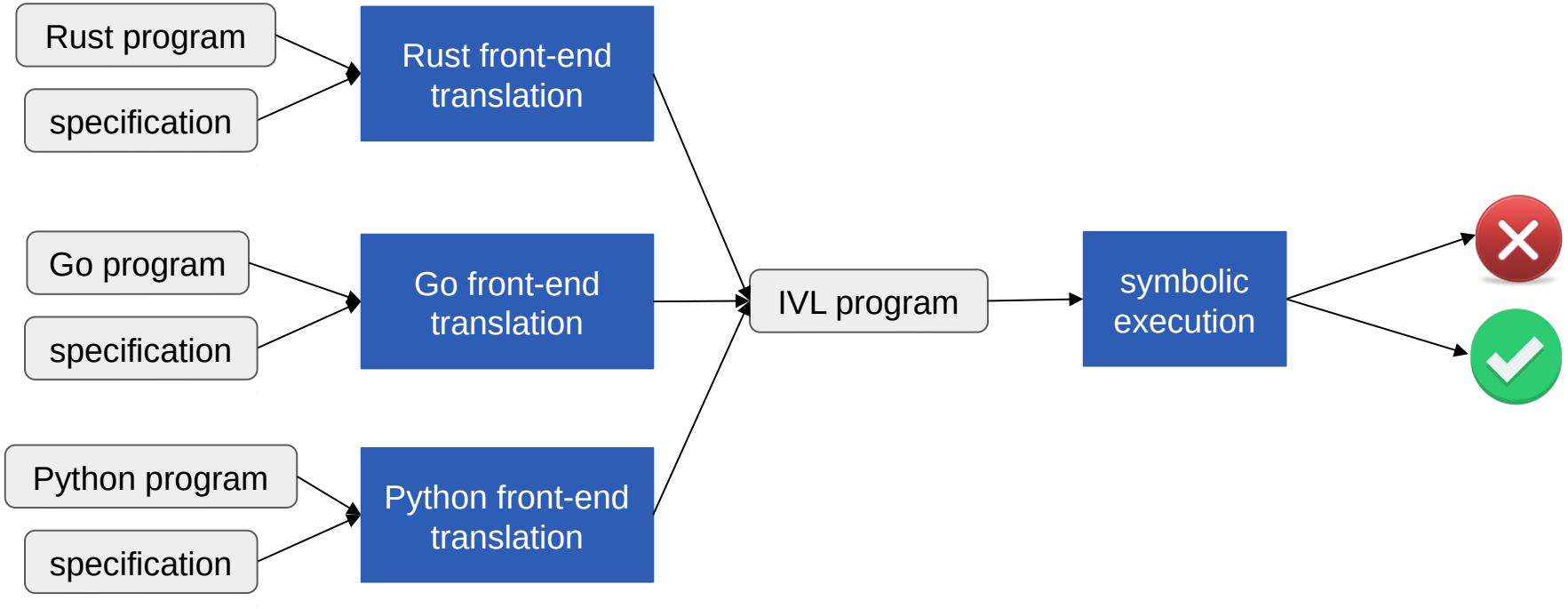
# Translational Separation Logic (SL) Verifiers



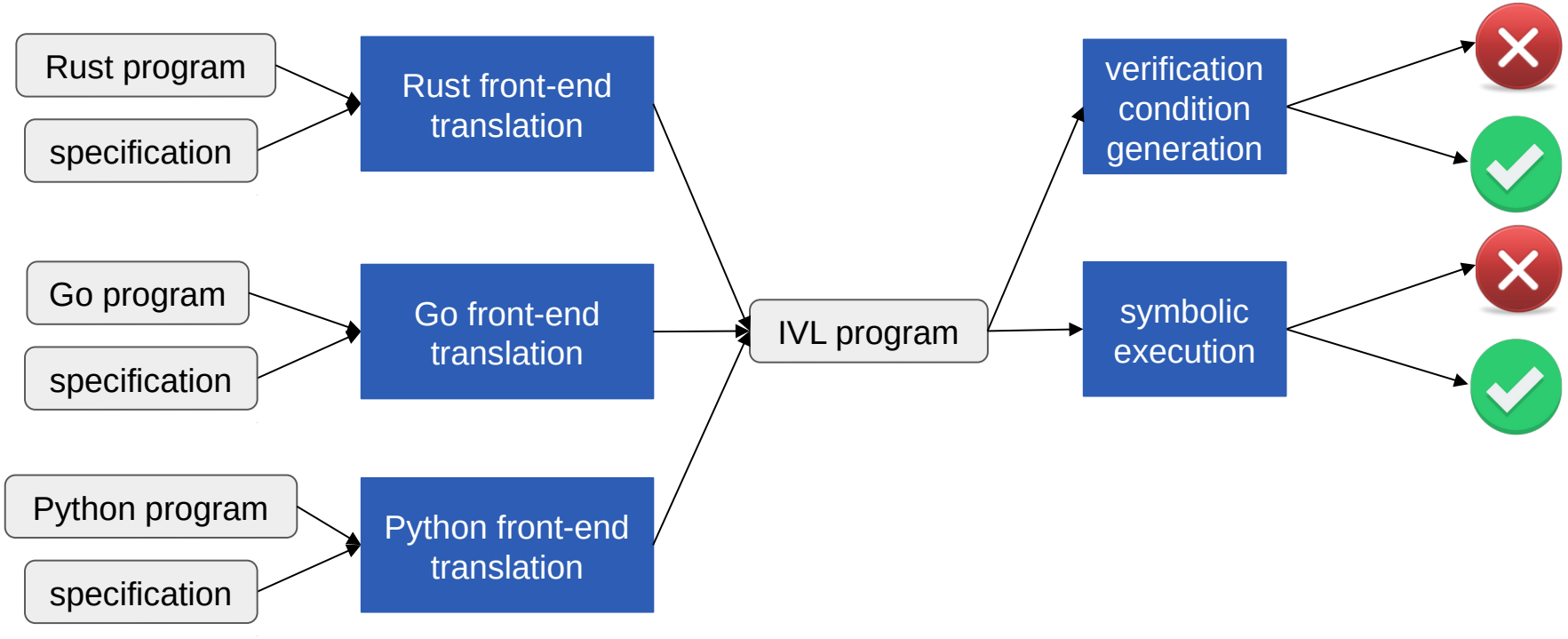
# Translational Separation Logic (SL) Verifiers



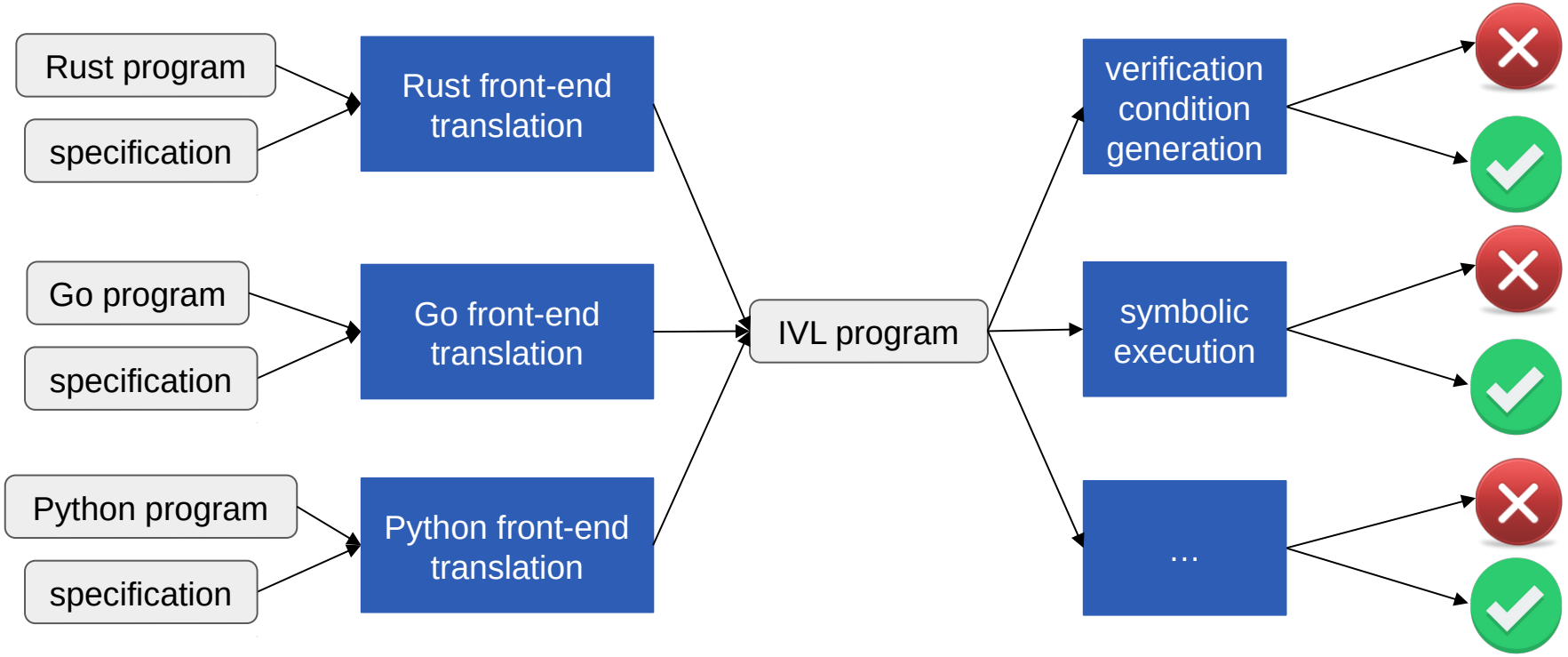
# Translational Separation Logic (SL) Verifiers



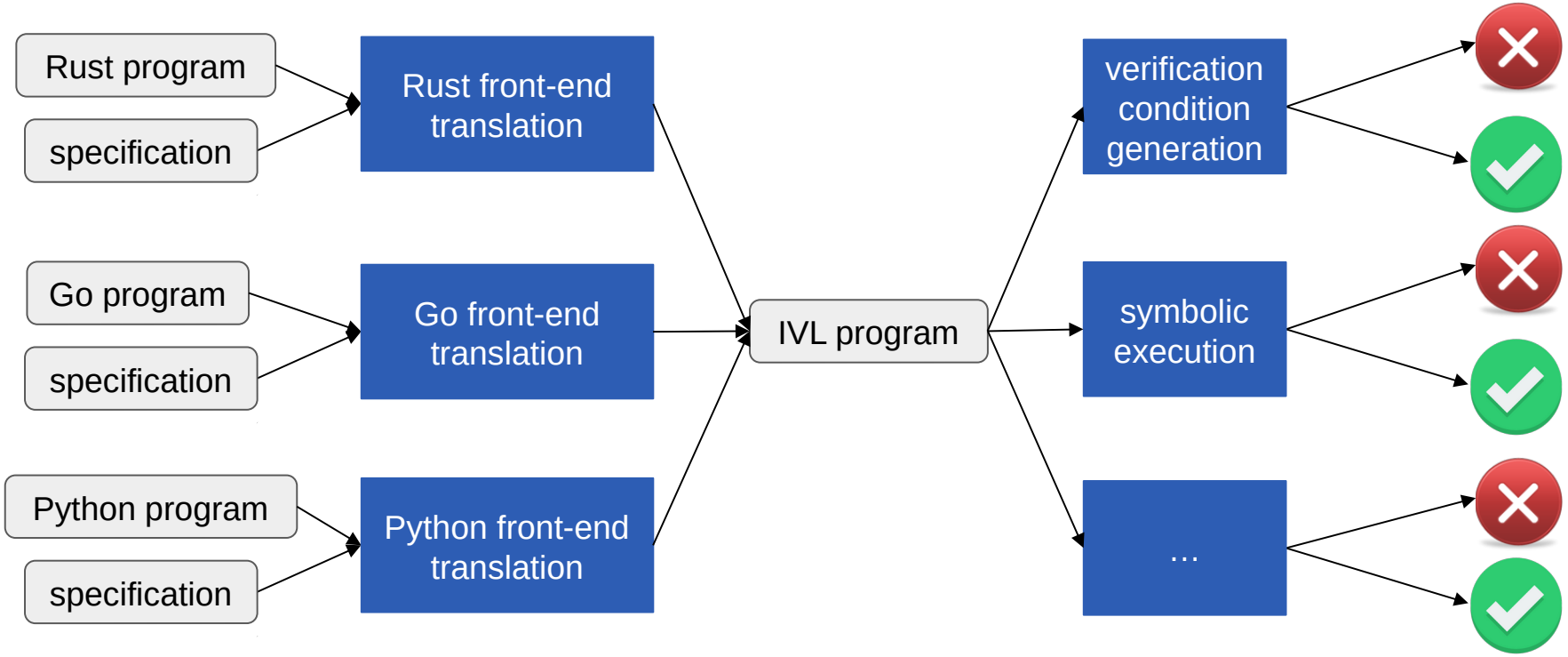
# Translational Separation Logic (SL) Verifiers



# Translational Separation Logic (SL) Verifiers



# Translational Separation Logic (SL) Verifiers

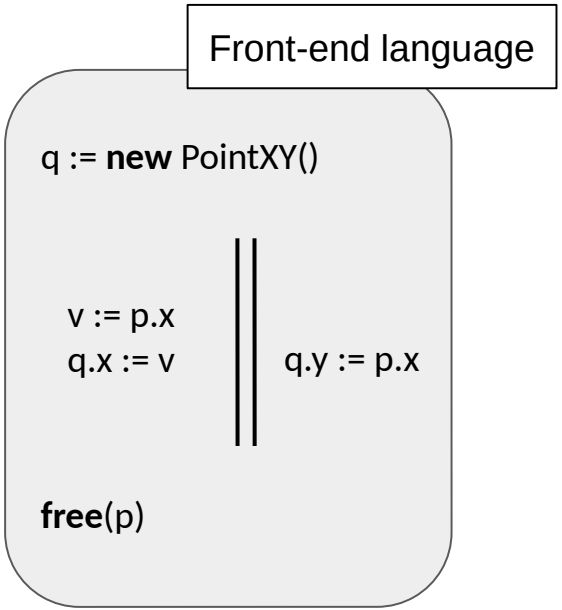


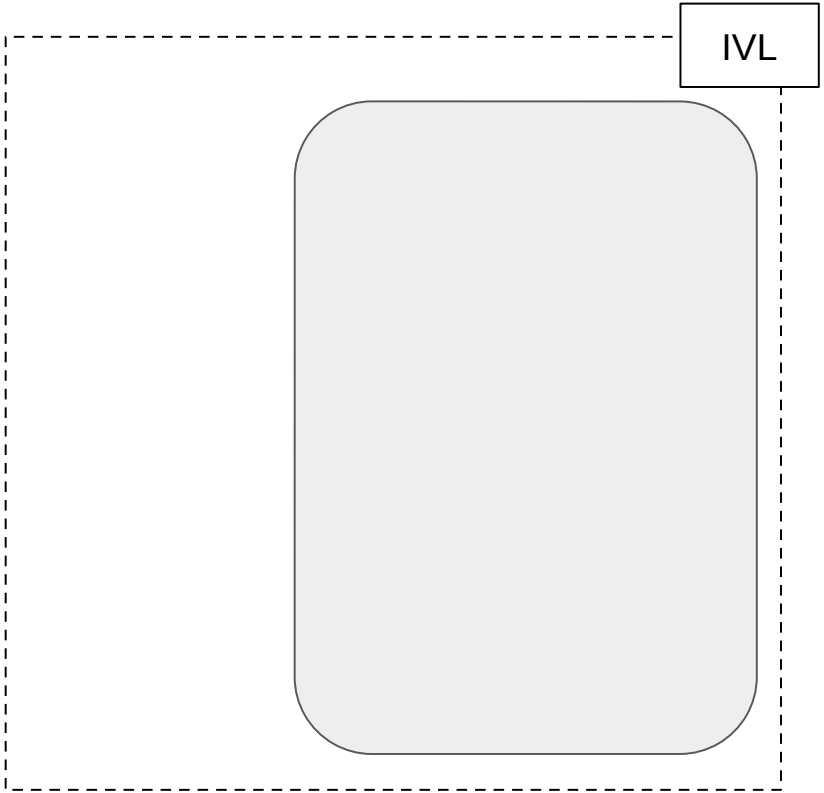

Gillian

VeriFast

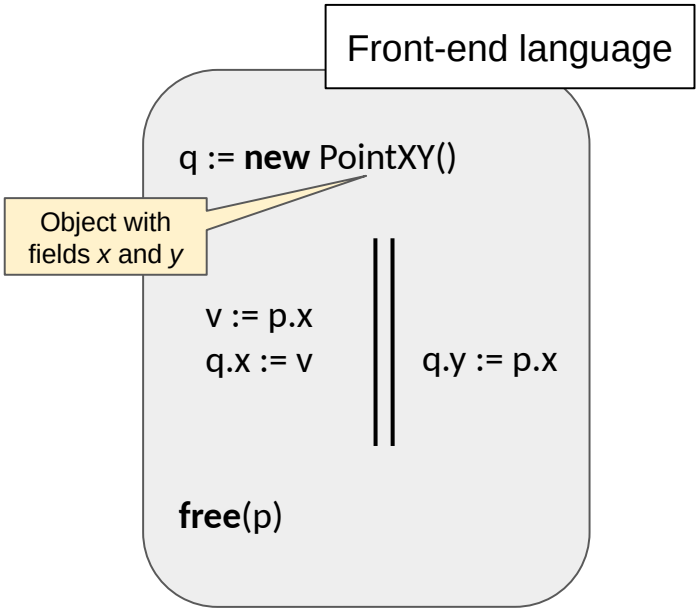
VIPER

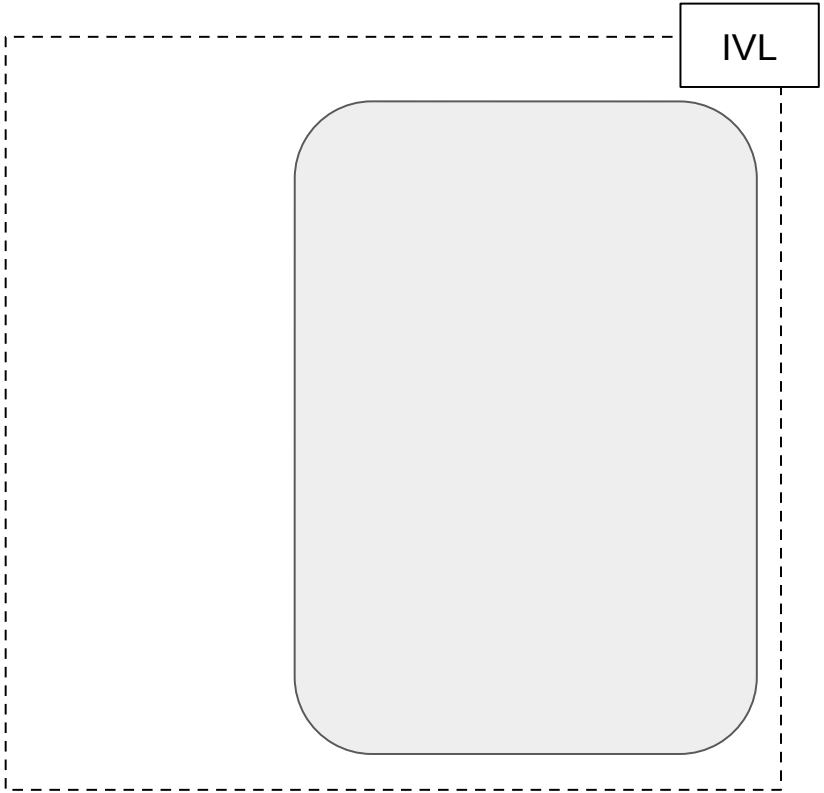
# Example: Encoding a Simple Parallel Program



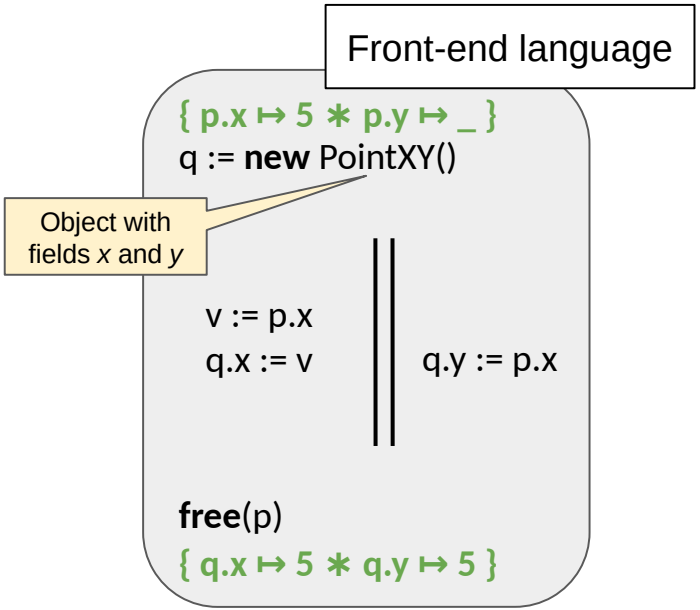



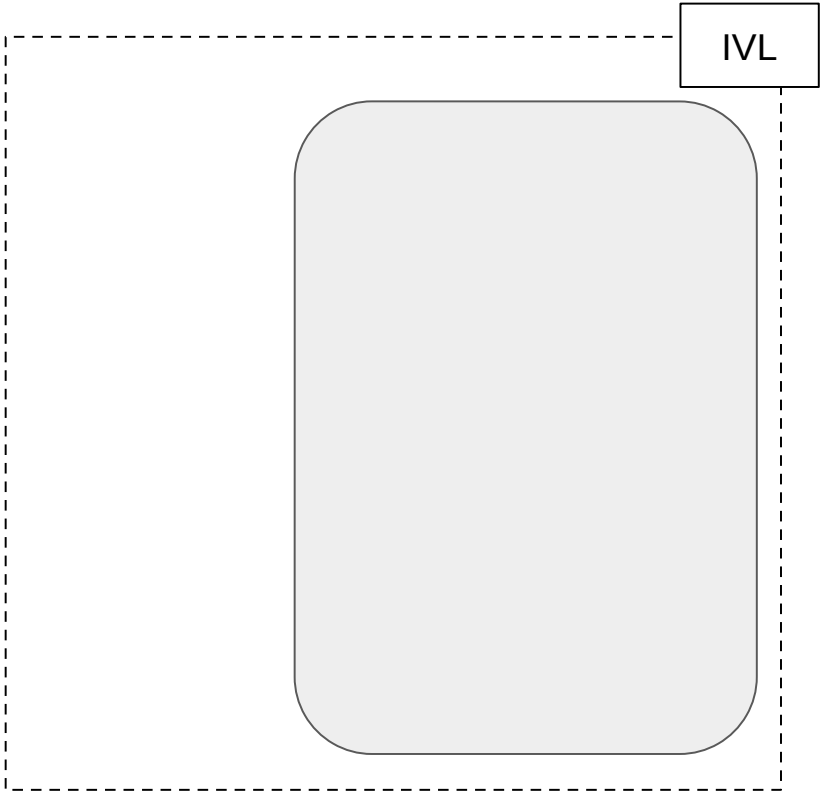
# Example: Encoding a Simple Parallel Program



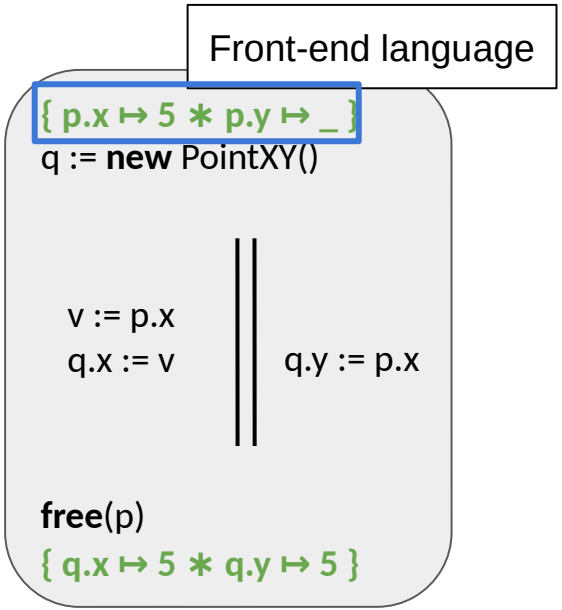



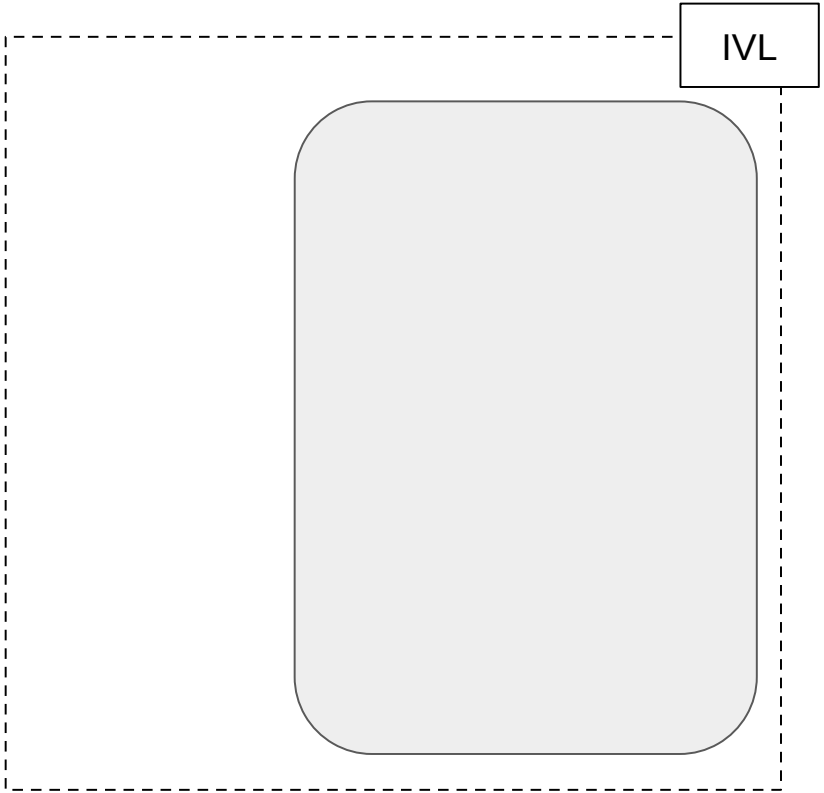
# Example: Encoding a Simple Parallel Program



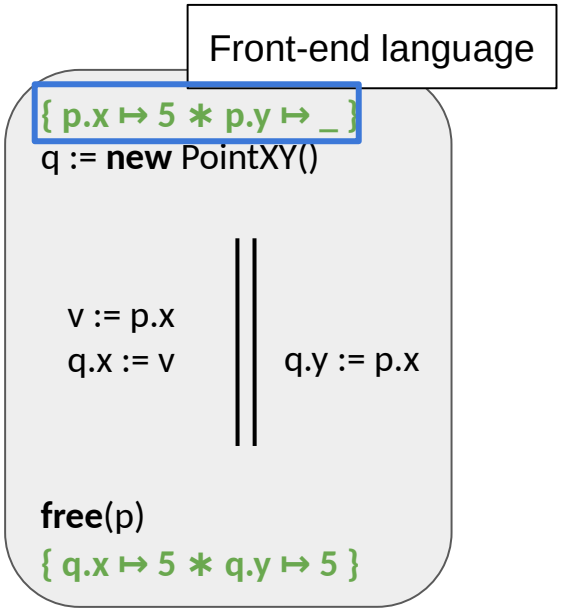



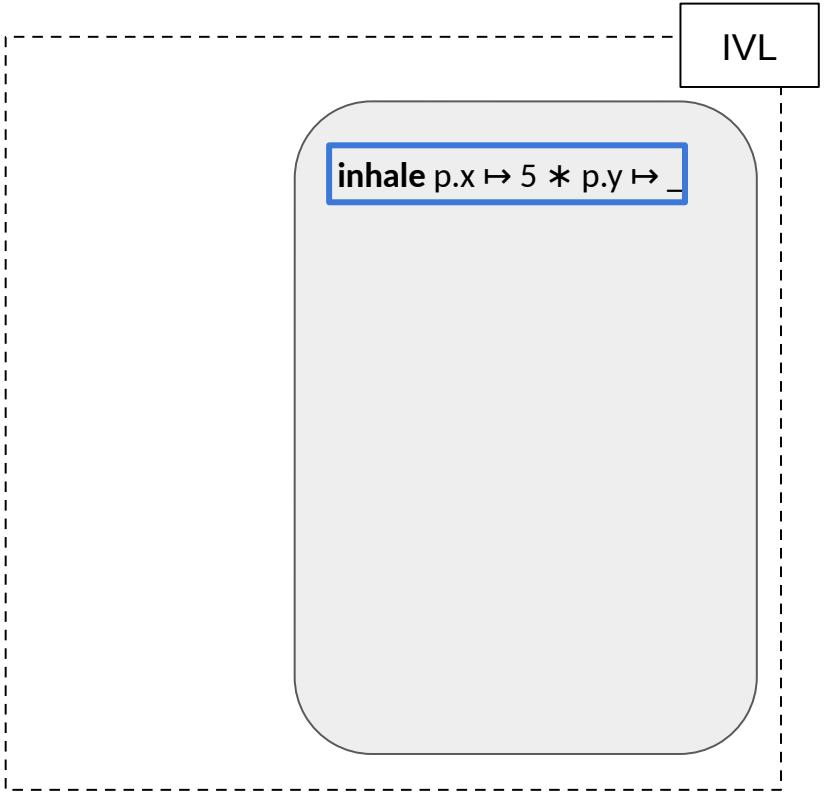
# Example: Encoding a Simple Parallel Program



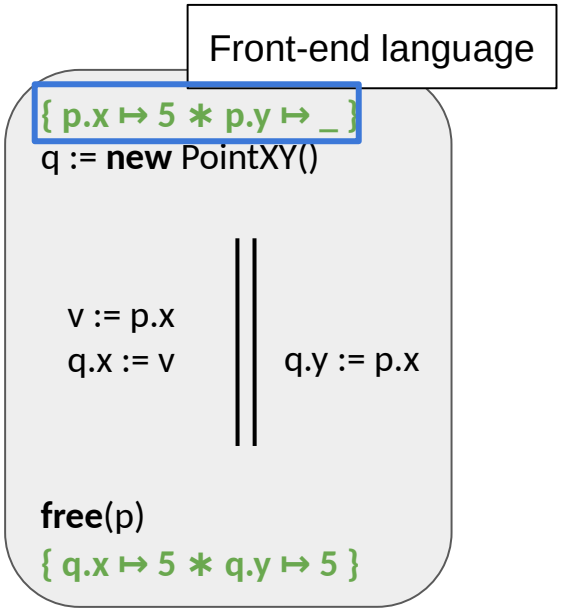



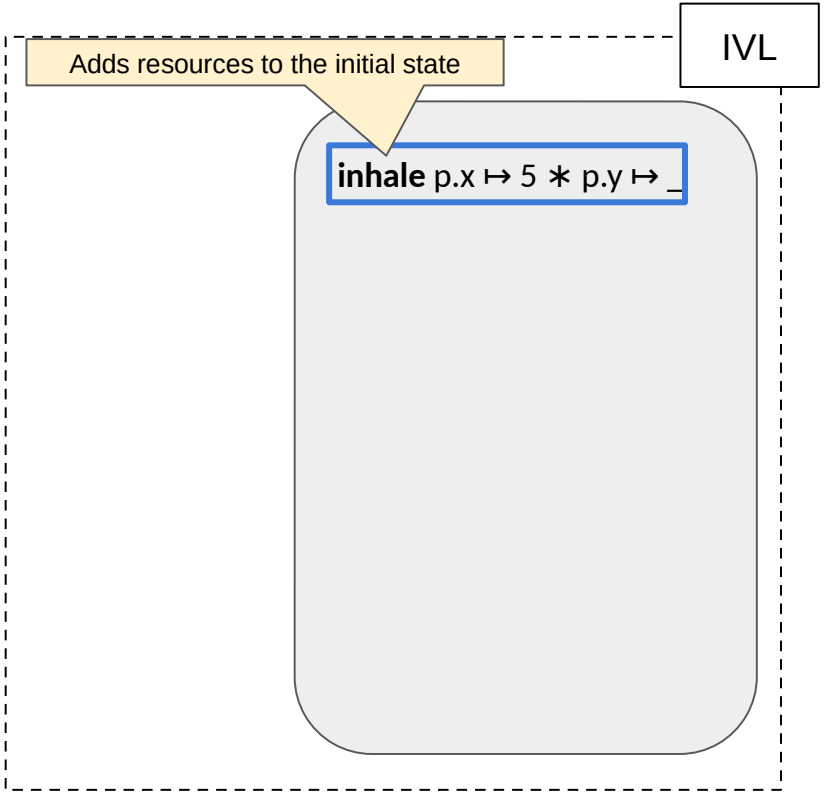
# Example: Encoding a Simple Parallel Program



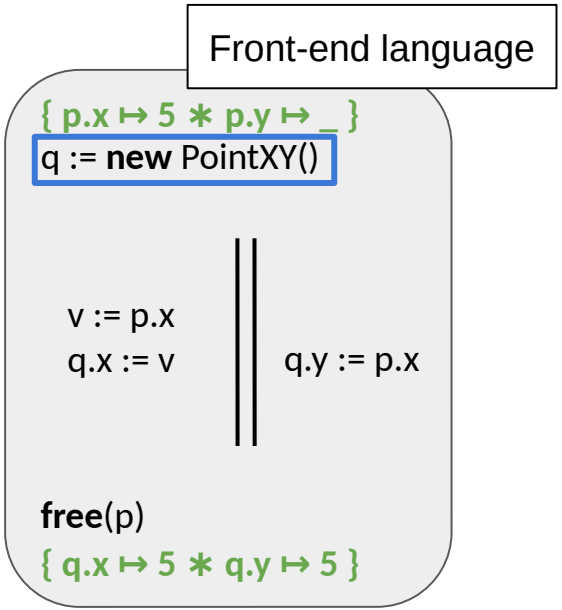



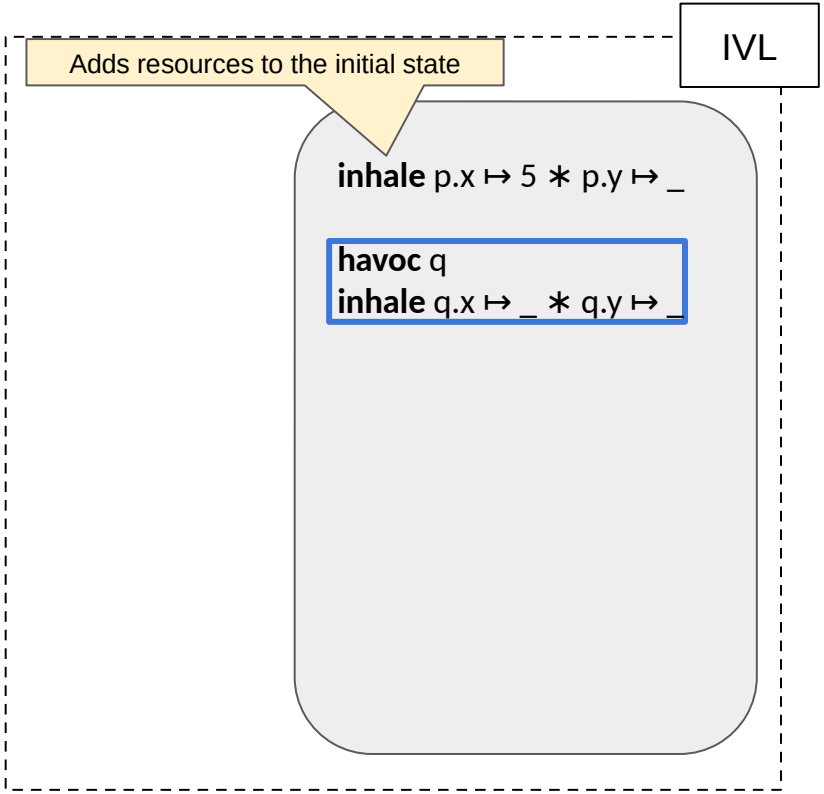
# Example: Encoding a Simple Parallel Program



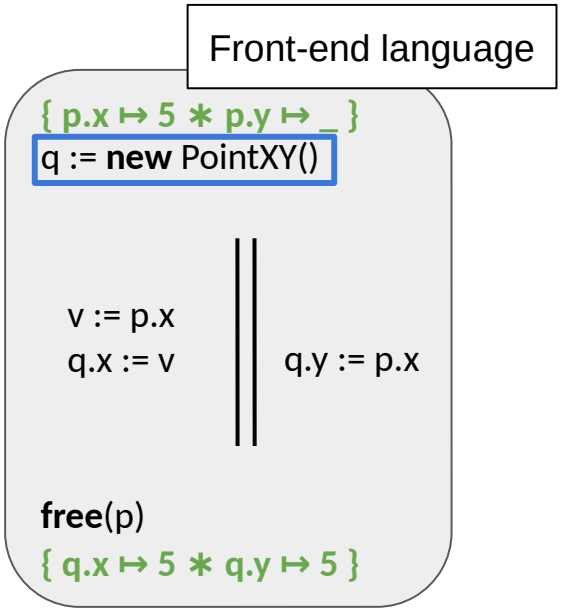



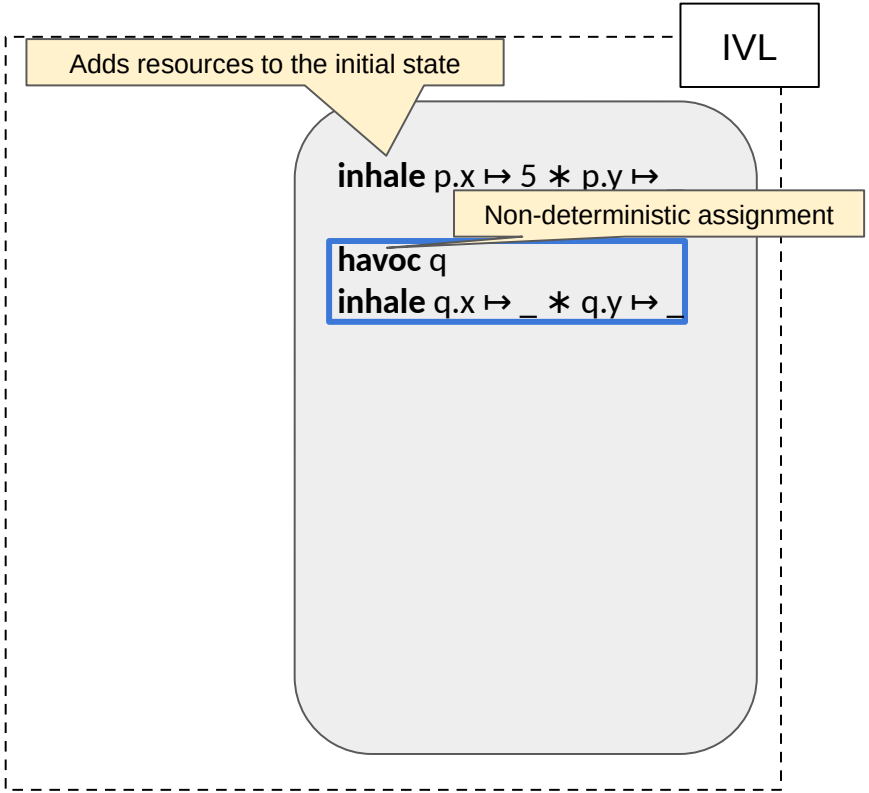
# Example: Encoding a Simple Parallel Program



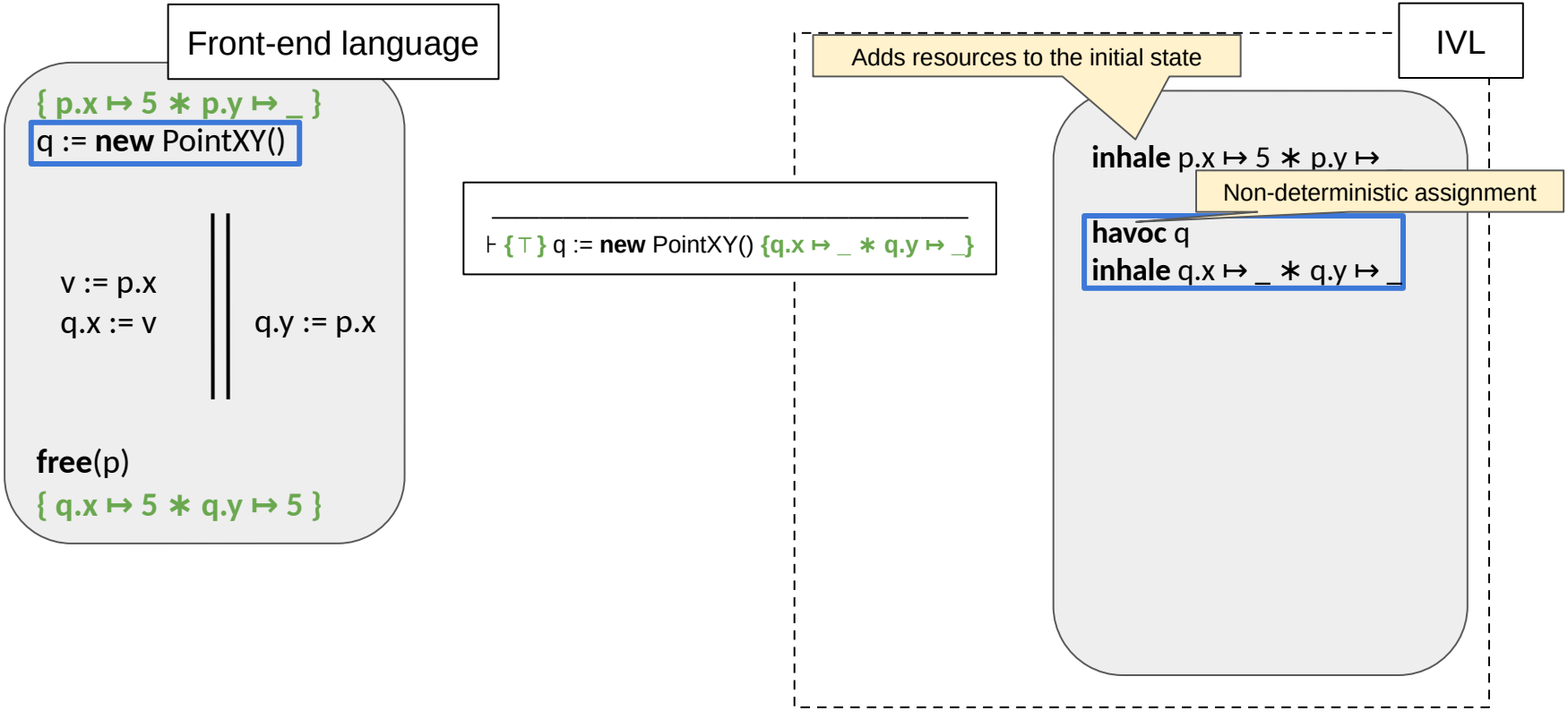



# Example: Encoding a Simple Parallel Program

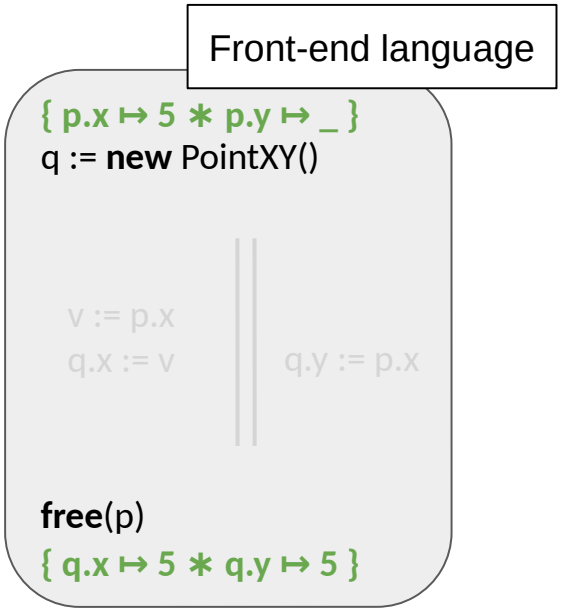


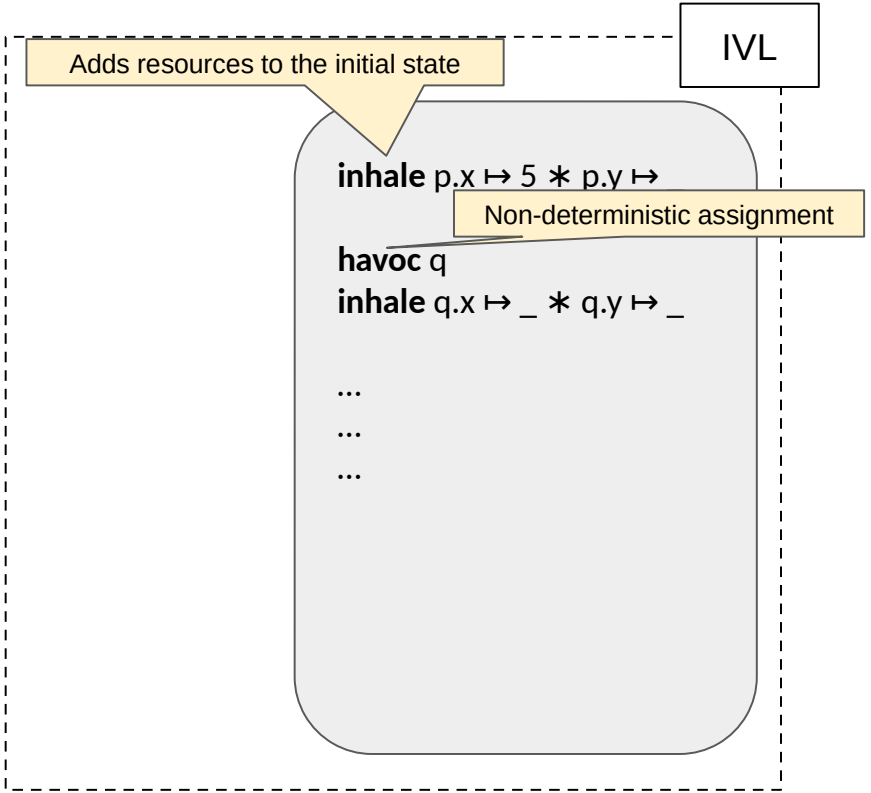



# Example: Encoding a Simple Parallel Program

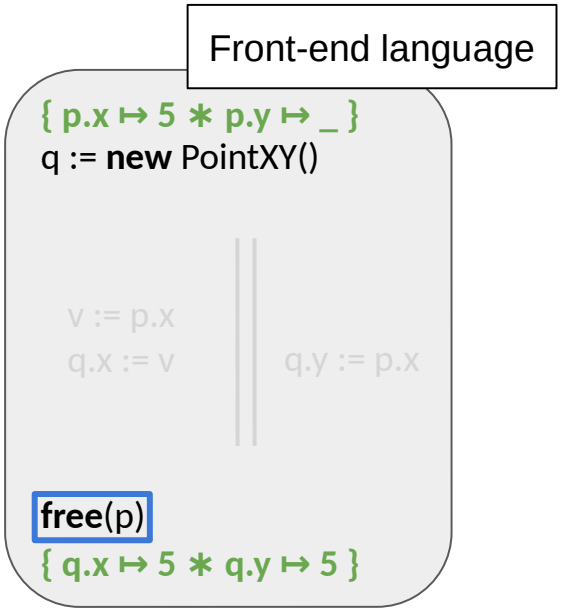


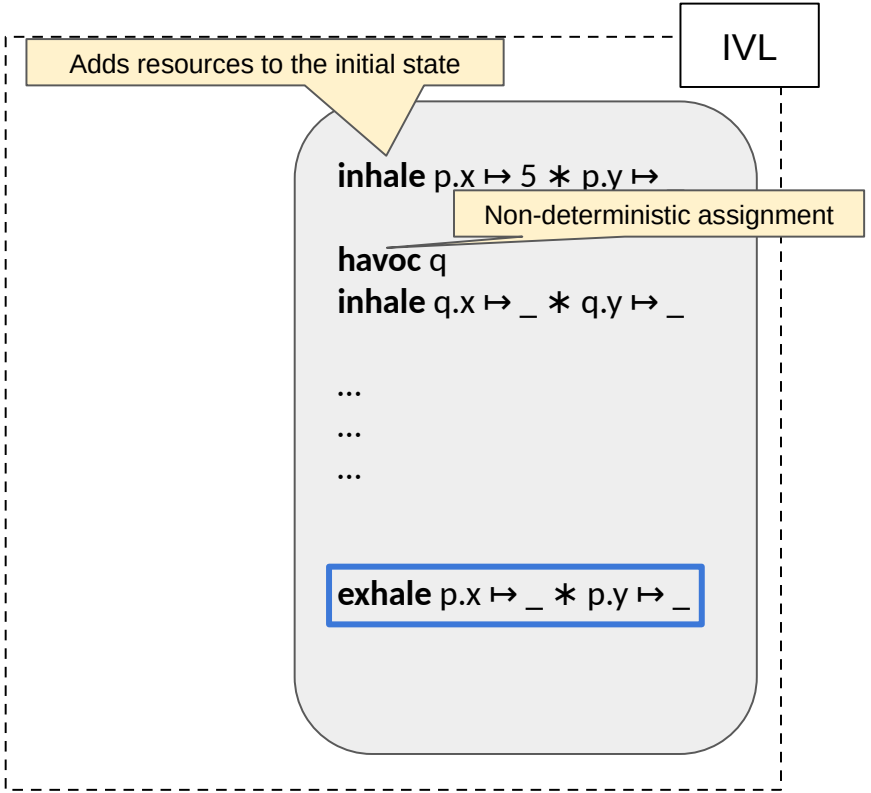
# Example: Encoding a Simple Parallel Program



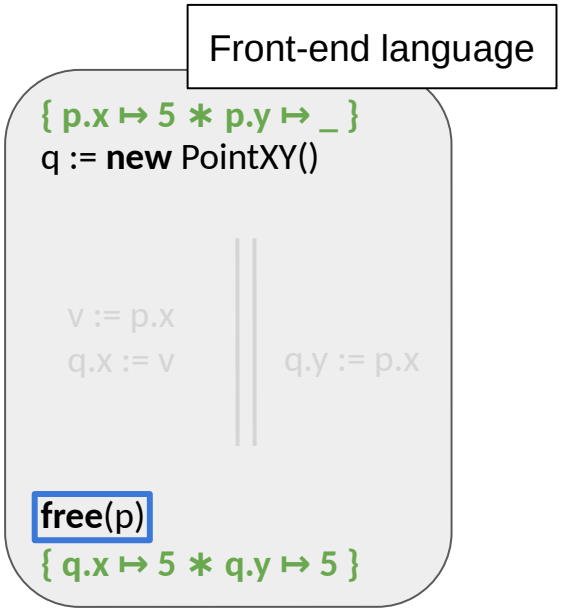



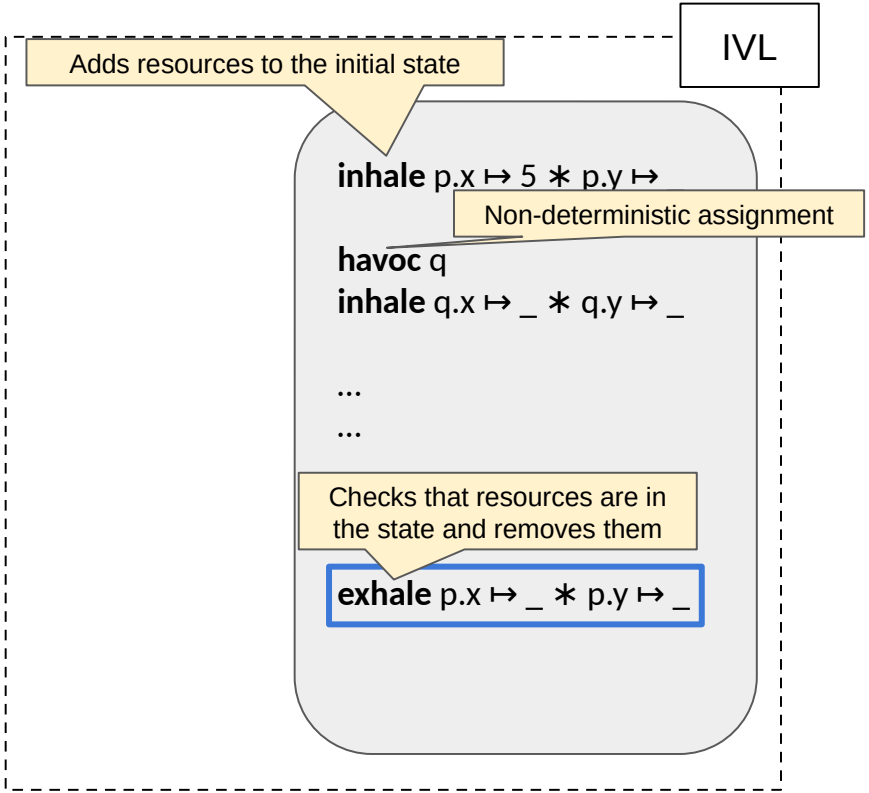
# Example: Encoding a Simple Parallel Program



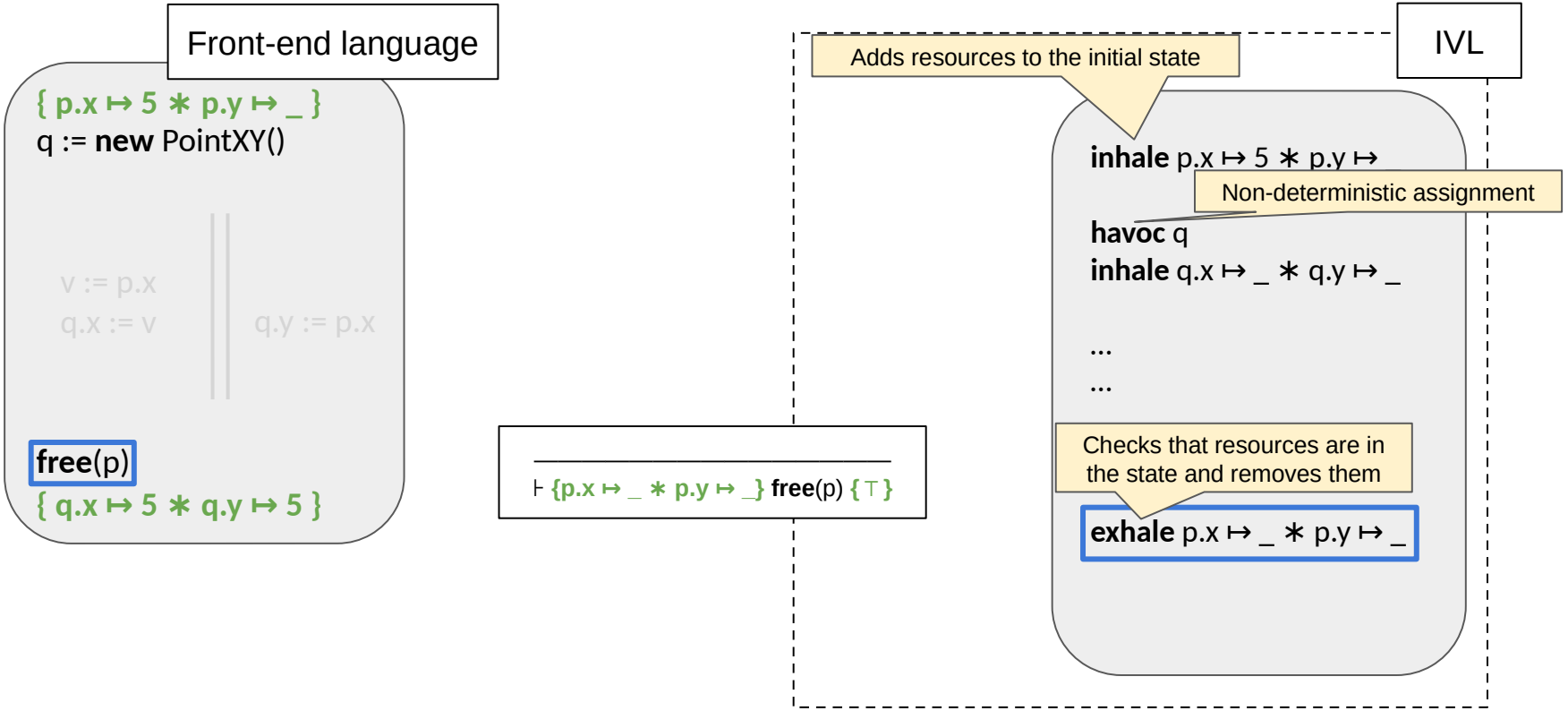



# Example: Encoding a Simple Parallel Program

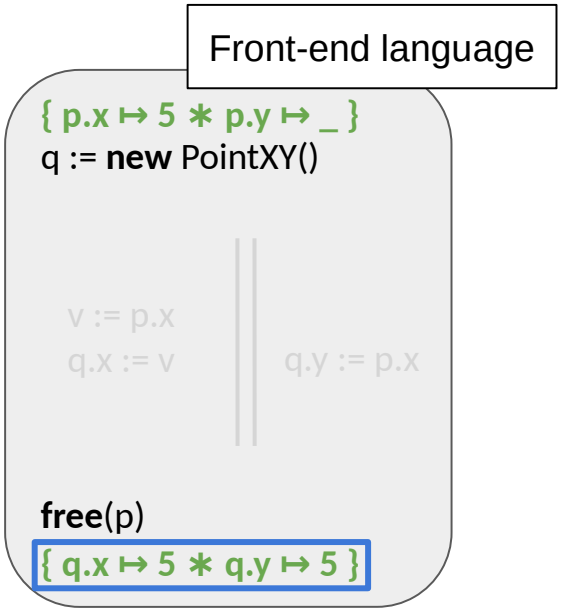


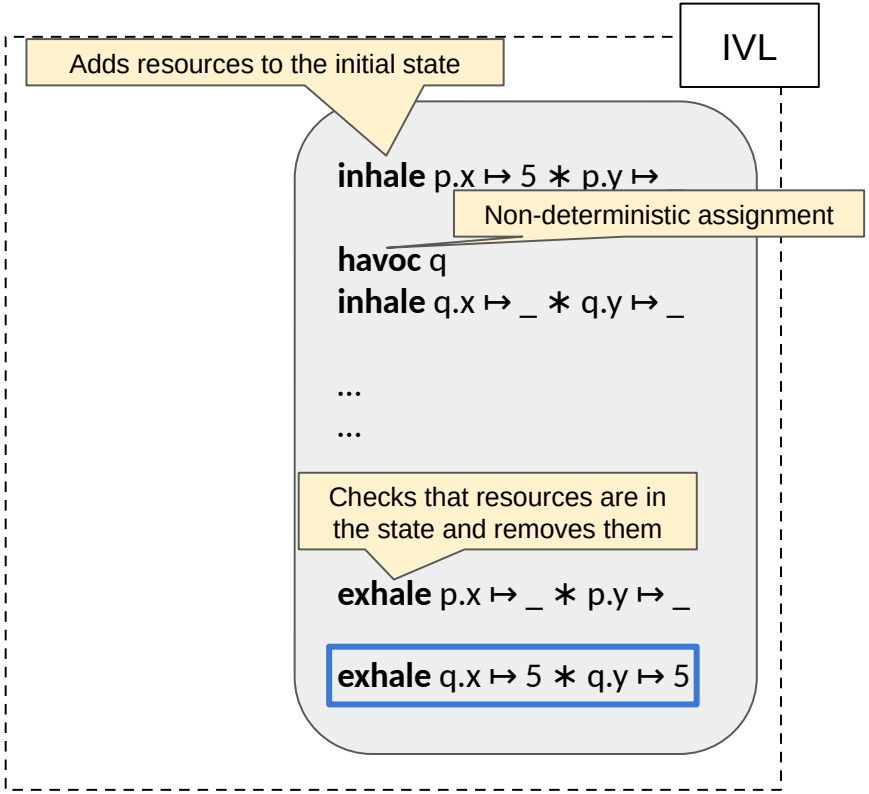



# Example: Encoding a Simple Parallel Program

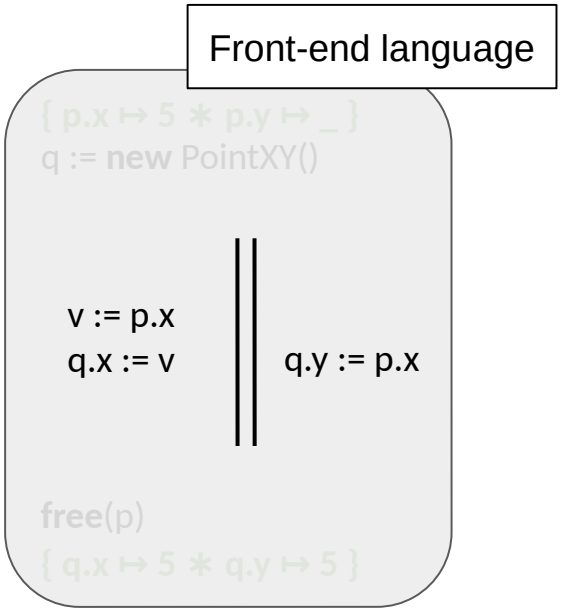


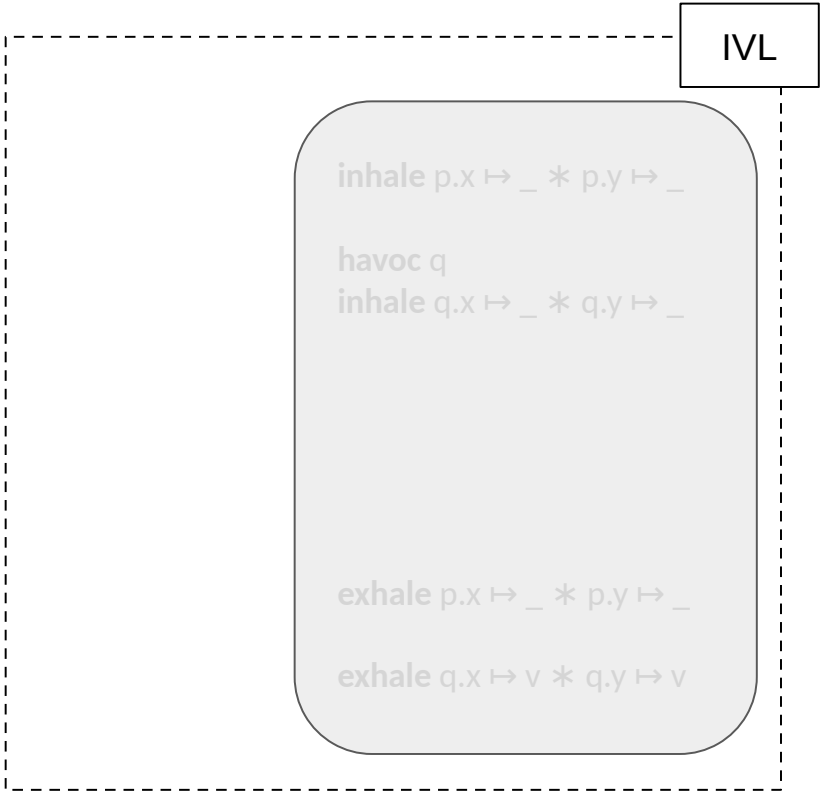
# Example: Encoding a Simple Parallel Program



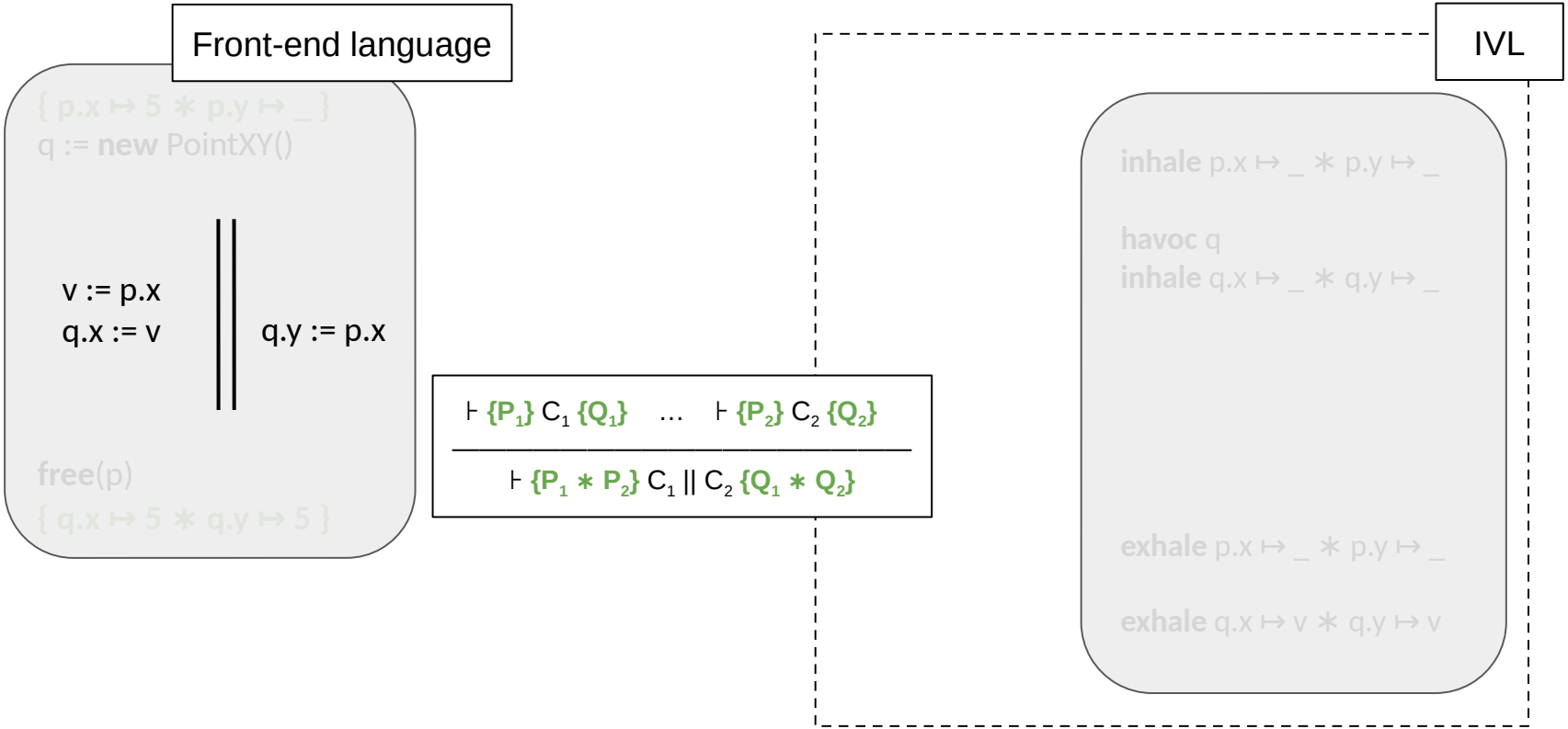



# Example: Encoding a Simple Parallel Program

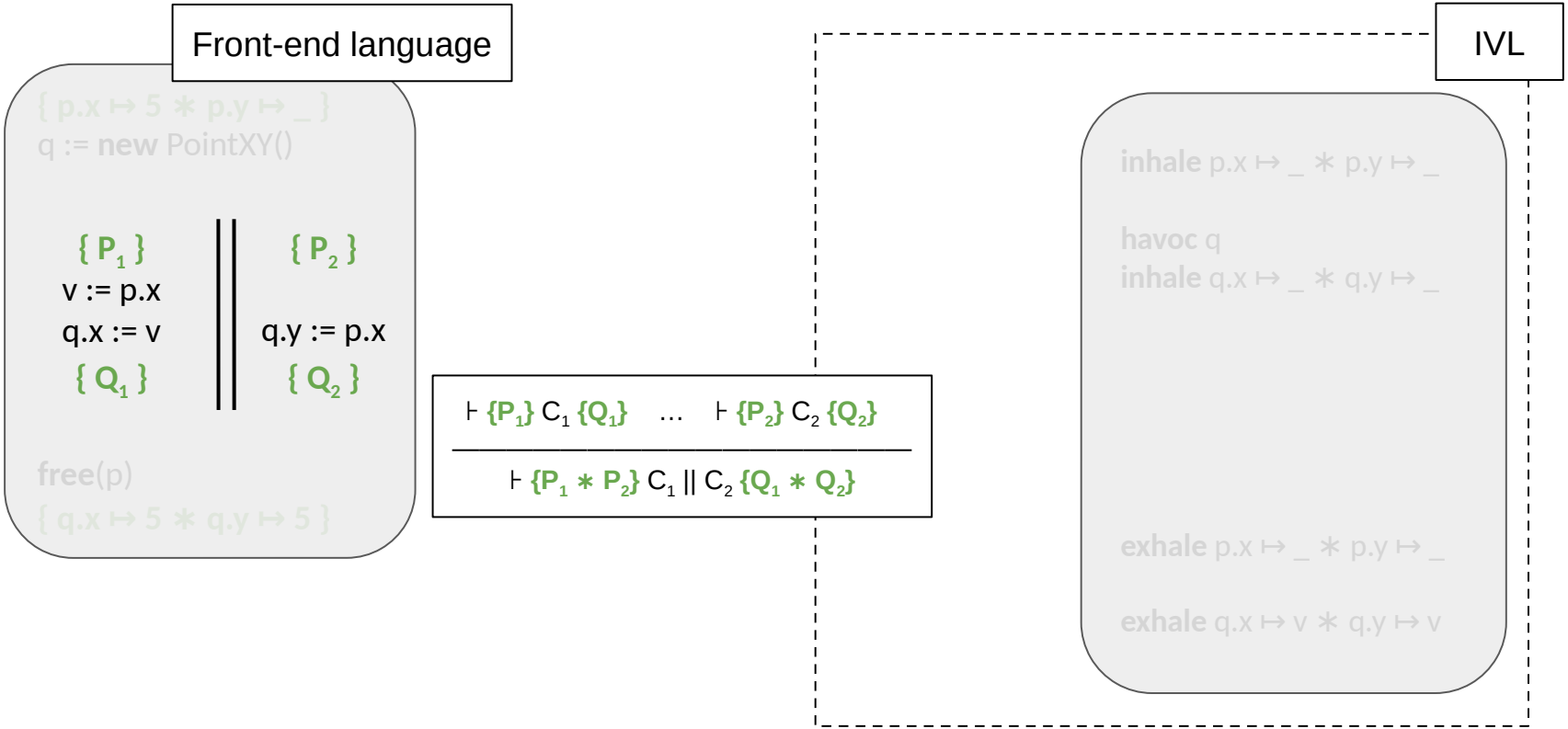


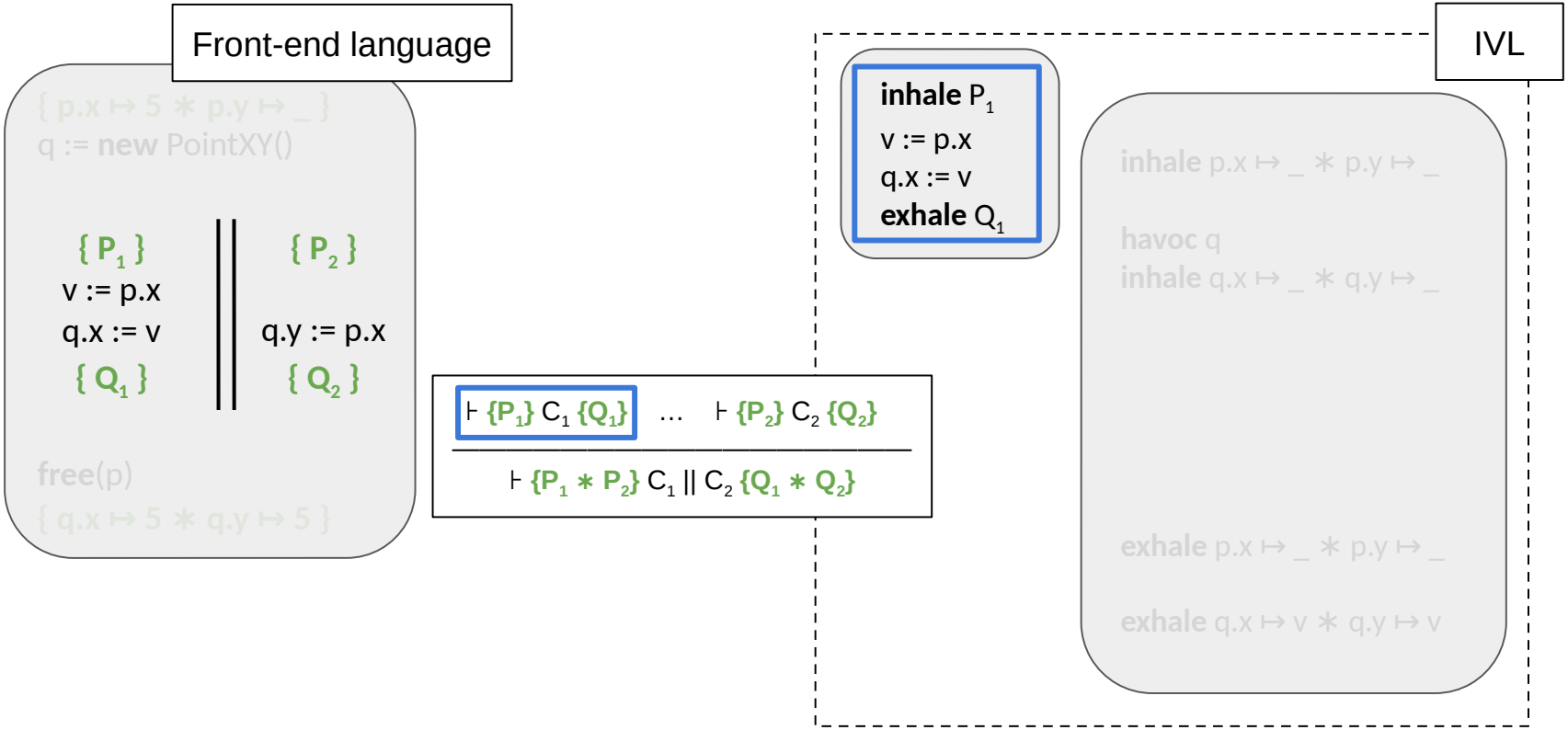
# Example: Encoding a Simple Parallel Program



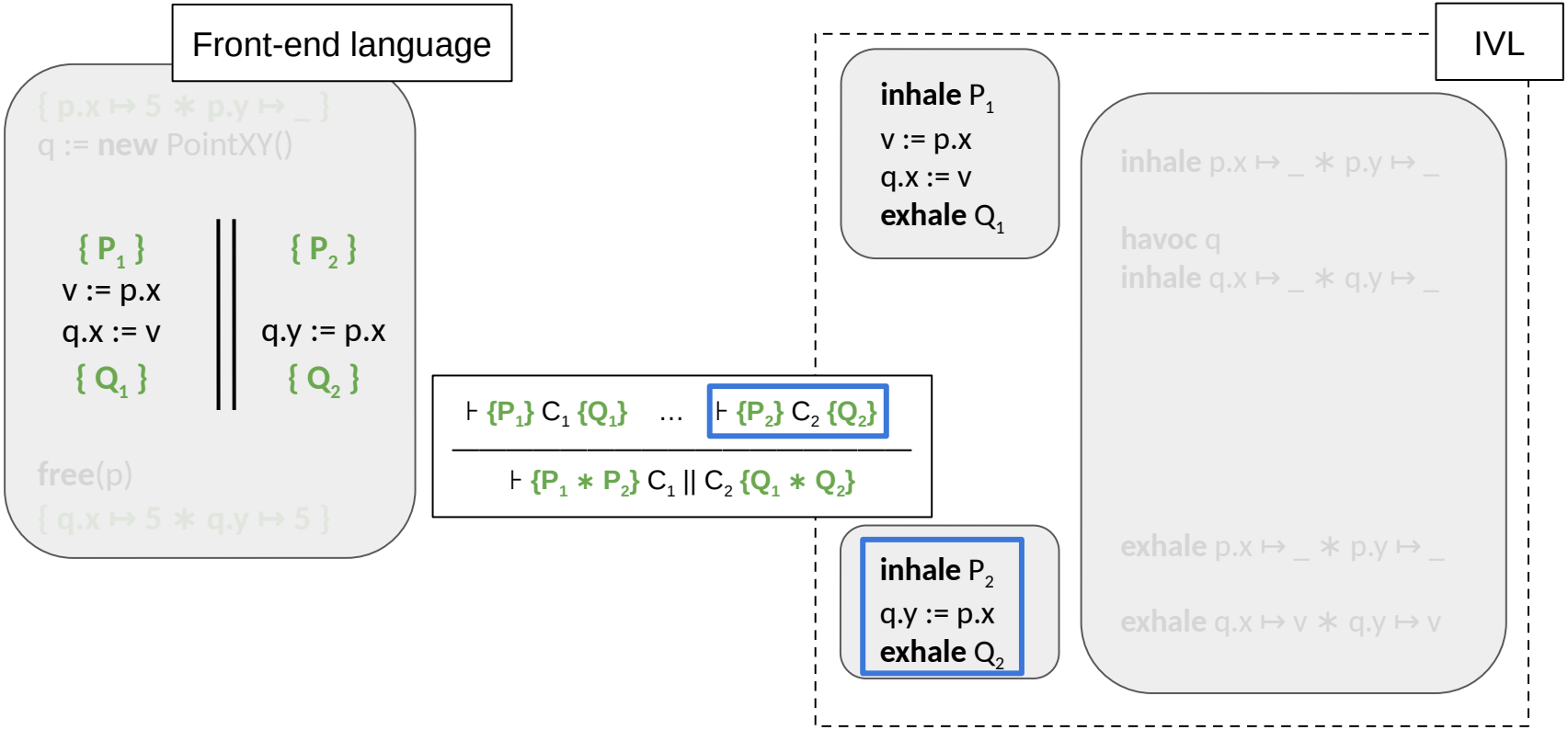
# Example: Encoding a Simple Parallel Program



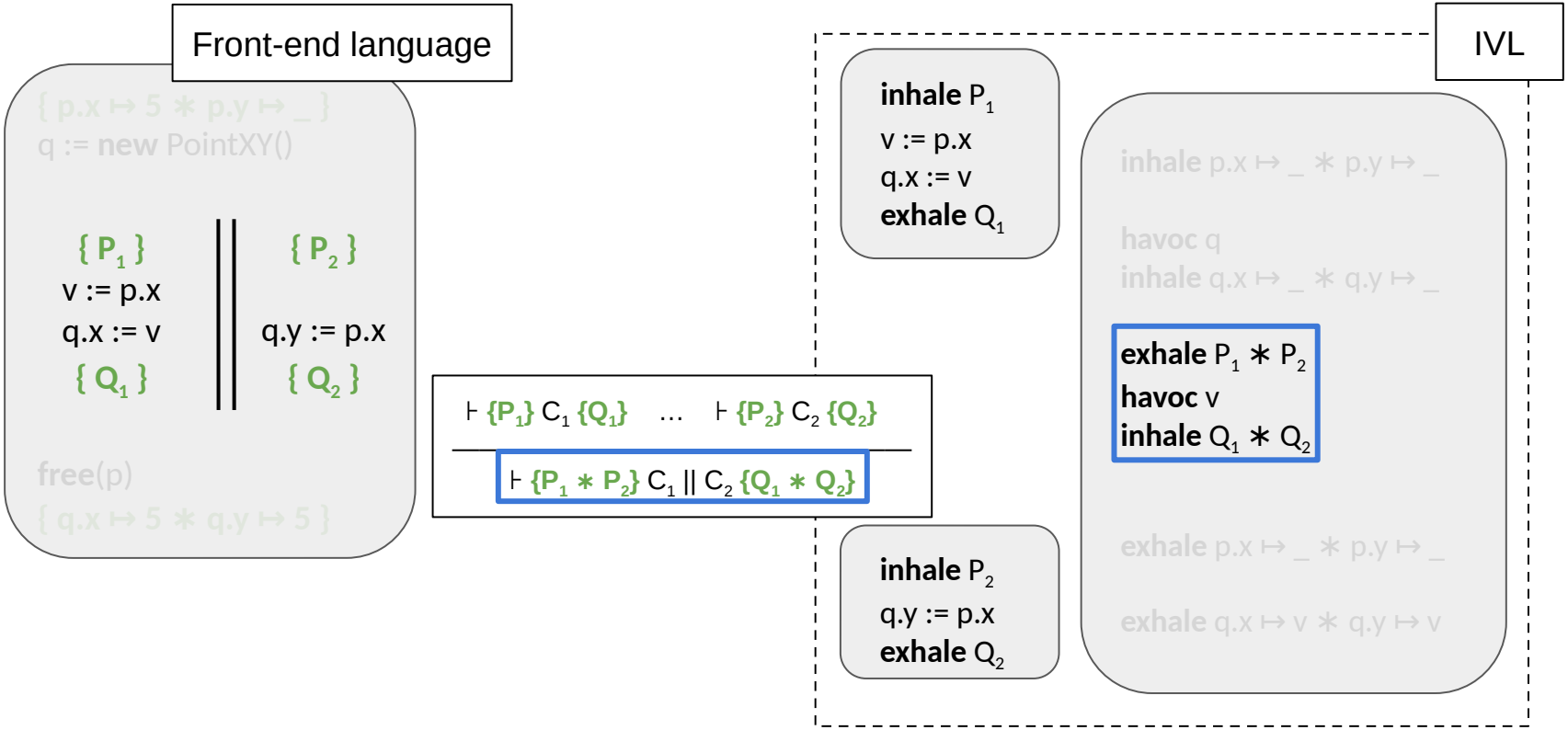
# Example: Encoding a Simple Parallel Program



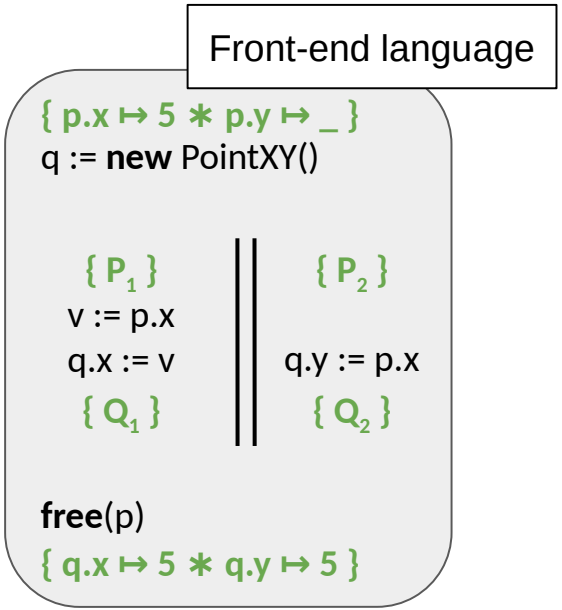
# Example: Encoding a Simple Parallel Program

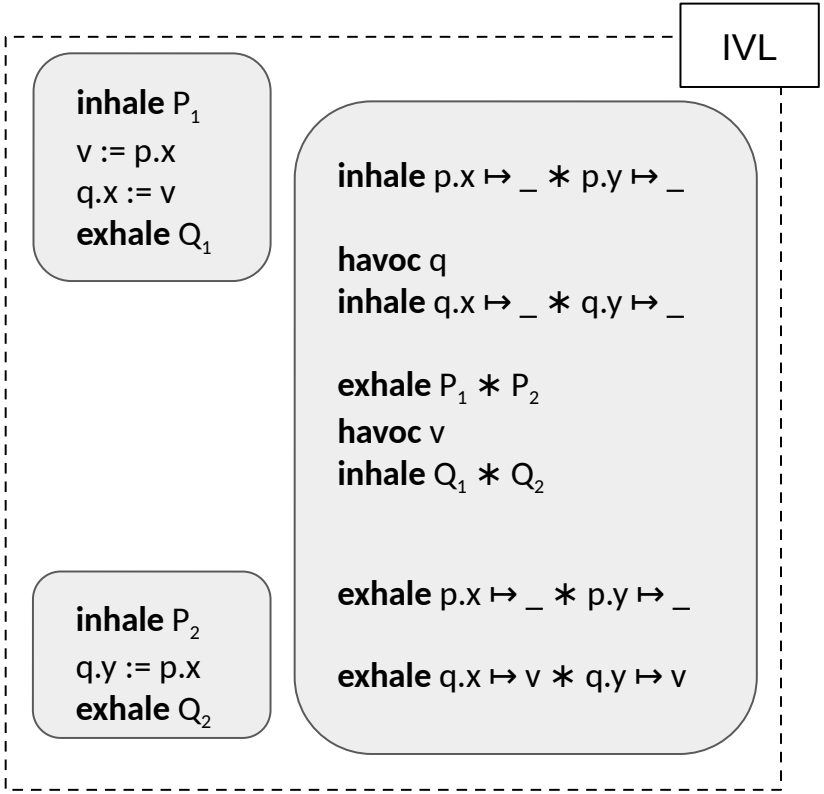


# Example: Encoding a Simple Parallel Program

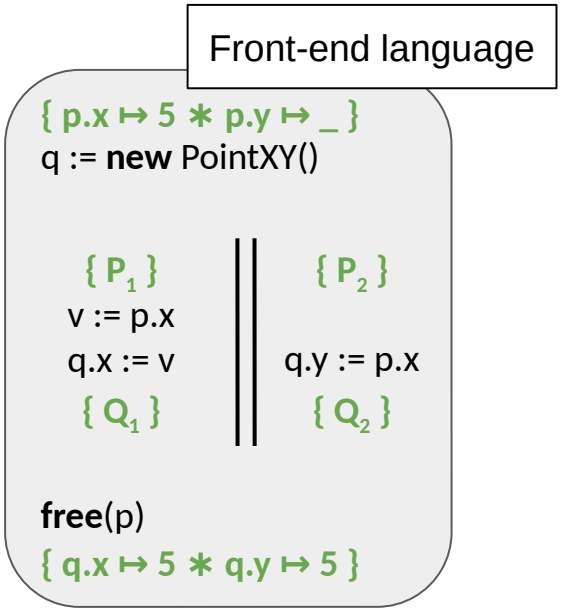


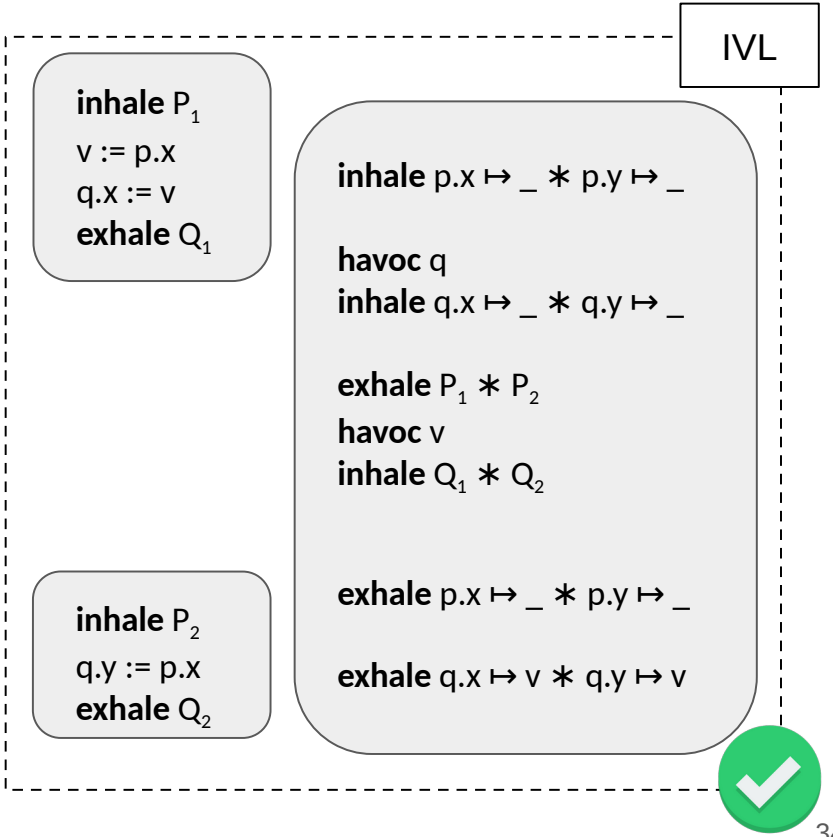
# Example: Encoding a Simple Parallel Program



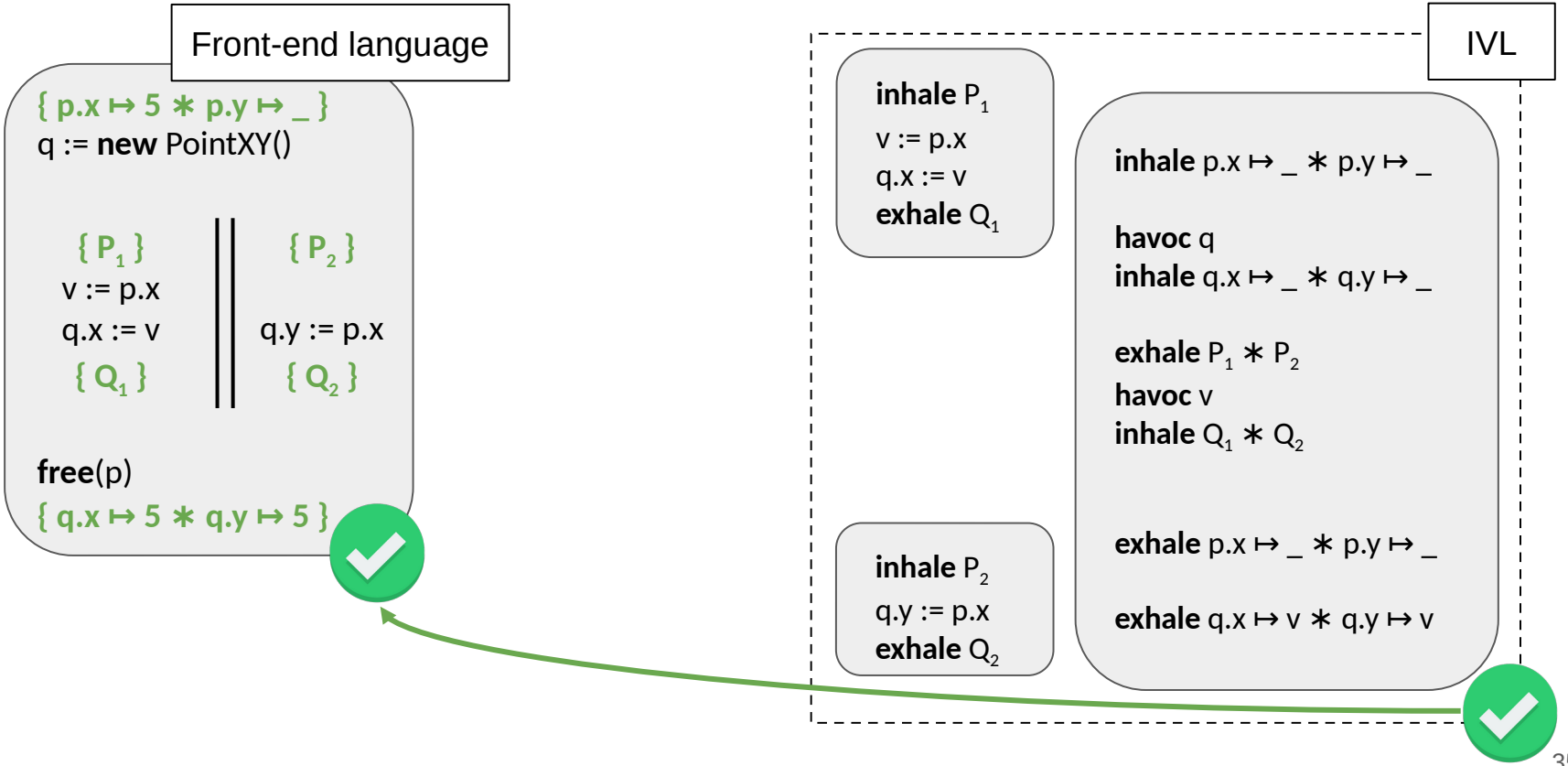



# Example: Encoding a Simple Parallel Program

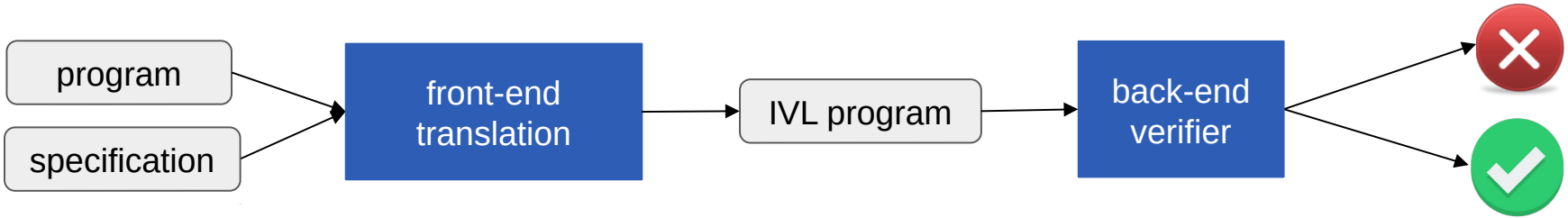


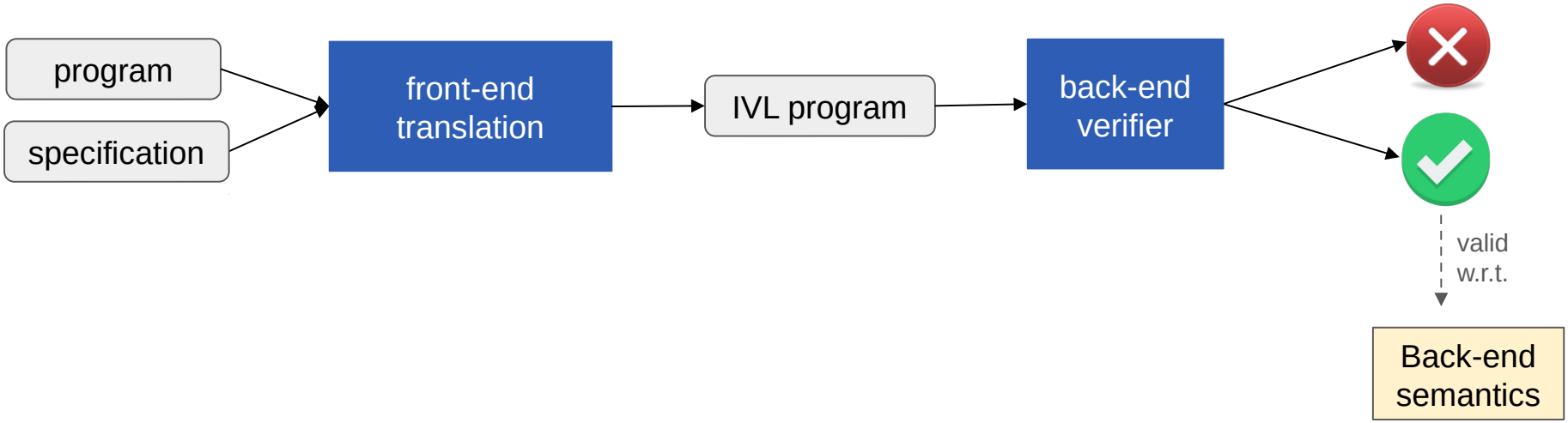
# Example: Encoding a Simple Parallel Program



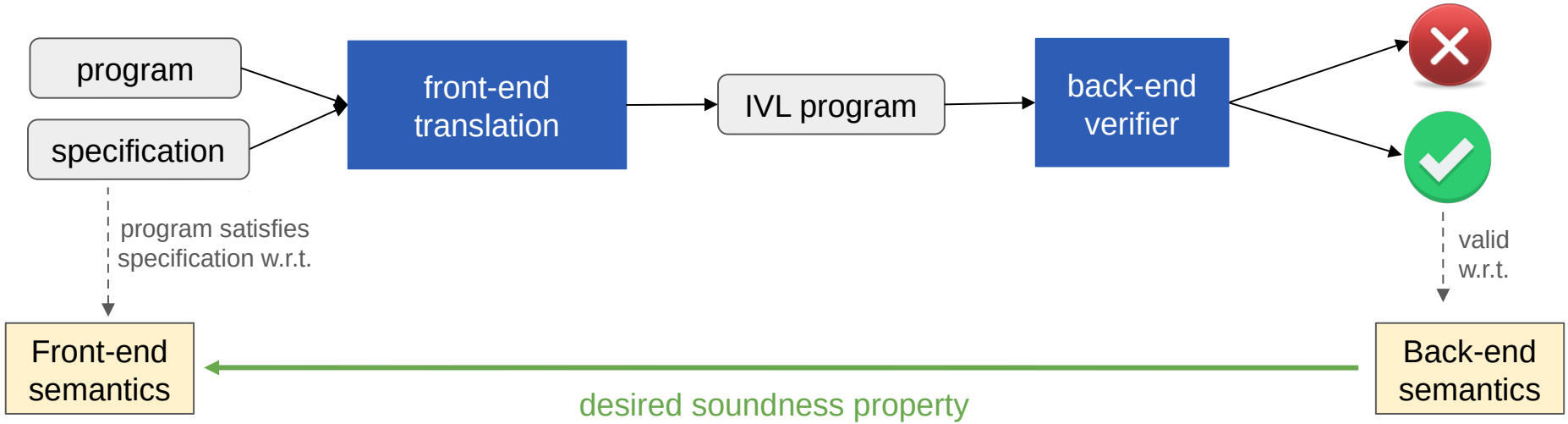
# Soundness of Translational Verifiers



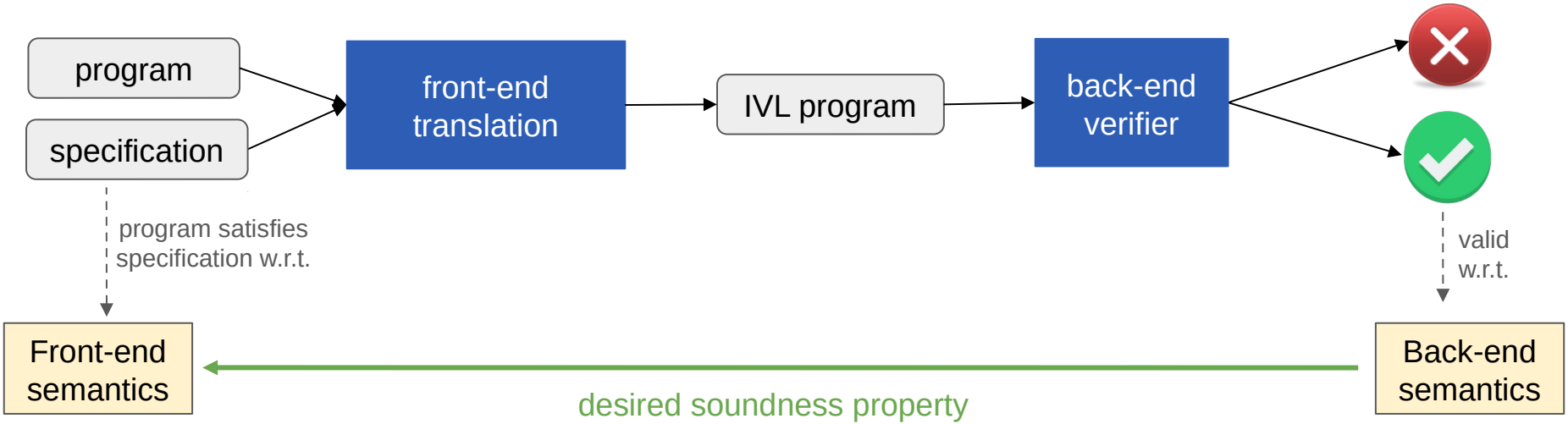
# Soundness of Translational Verifiers



# Soundness of Translational Verifiers

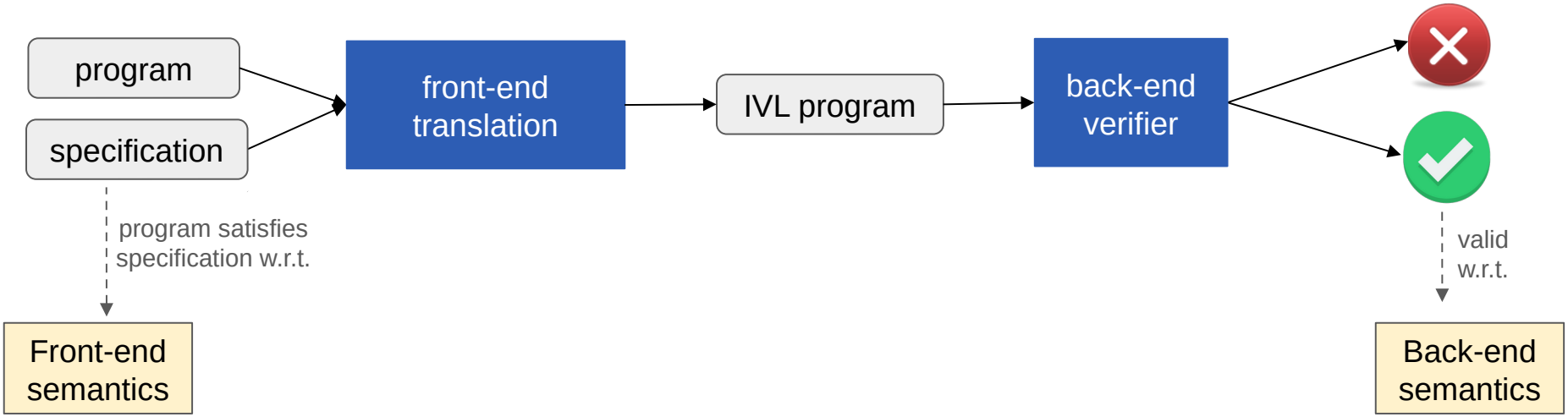


# Soundness of Translational Verifiers

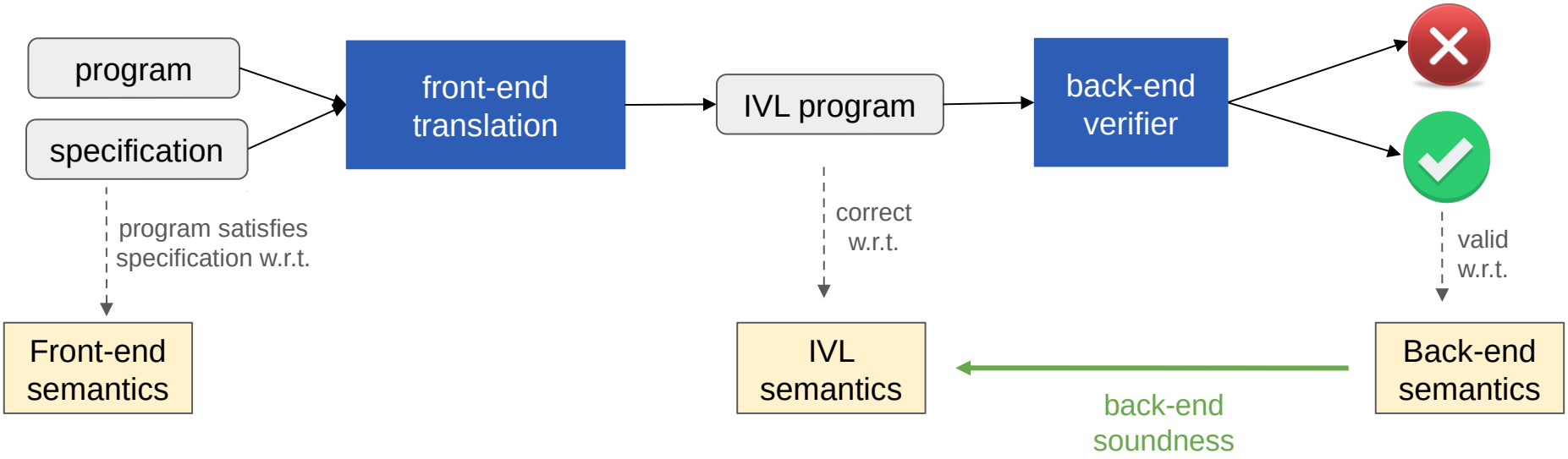



**Goal**: Developing a general framework for formally establishing the desired soundness property.

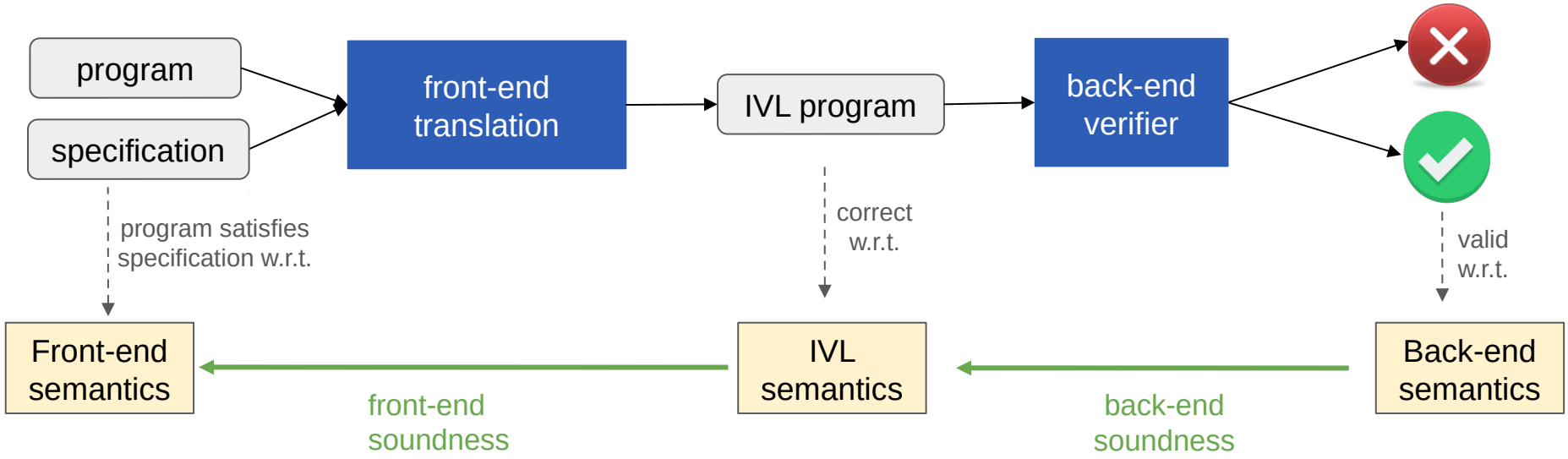
# Soundness of Translational Verifiers



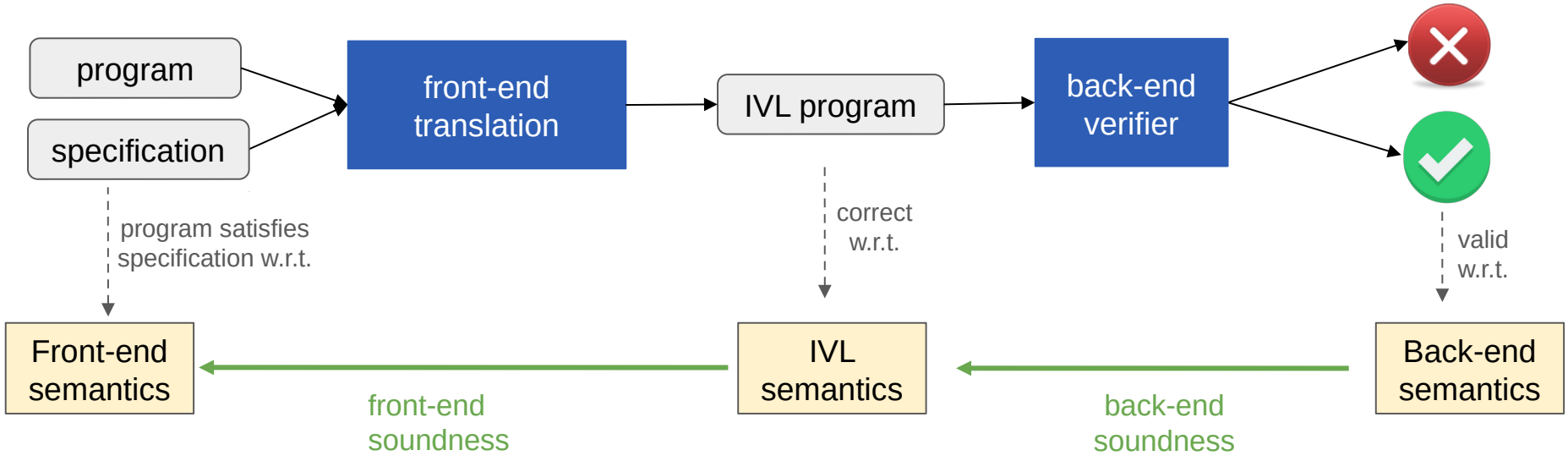
# Soundness of Translational Verifiers



# Soundness of Translational Verifiers

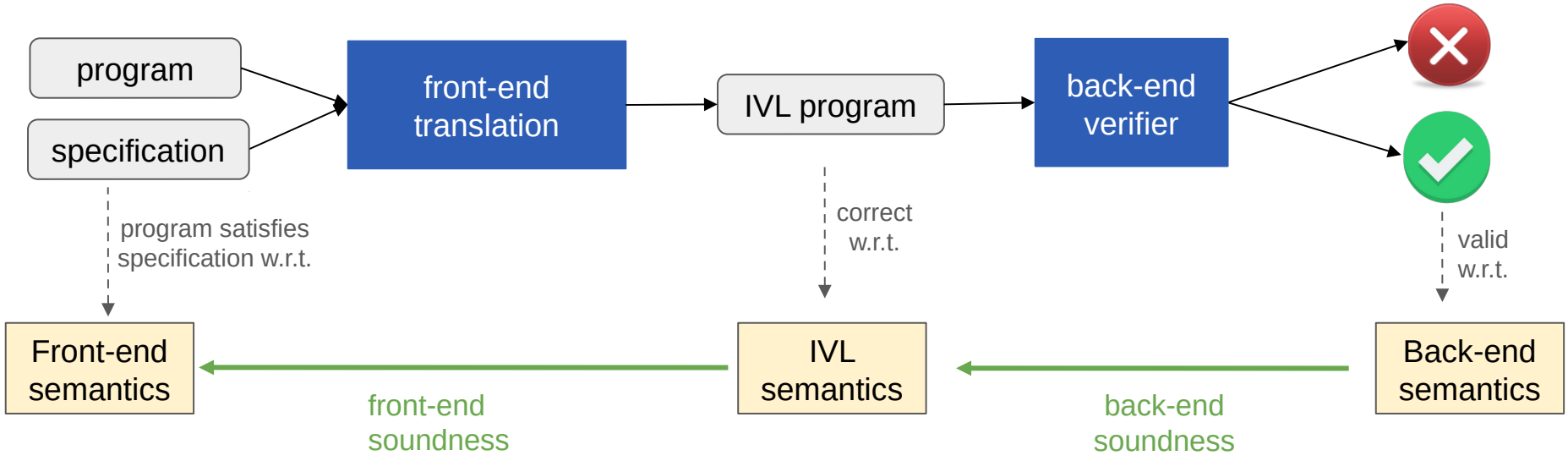


# Soundness of Translational Verifiers




**Challenge 1**:
IVL semantics

# Soundness of Translational Verifiers




**Challenge 3**: Front-end soundness

**Challenge 1**: IVL semantics

**Challenge 2**: Back-end soundness

# CoreIVL: A Core Language for SL-based IVLs


VeriFast

# CoreIVL: A Core Language for SL-based IVLs


VeriFast


$$C ::= \texttt{inhale}\ A \mid \texttt{exhale}\ A \mid \texttt{havoc}\ x \mid C;C \mid \texttt{if}(b)\ \{C\}\ \texttt{else}\ \{C\} \mid x := e \mid \texttt{skip} \mid \texttt{custom}\ K$$

# CoreIVL: A Core Language for SL-based IVLs


VeriFast


Adds resources

$$C ::= \texttt{inhale}\ A \mid \texttt{exhale}\ A \mid \texttt{havoc}\ x \mid C;C \mid \texttt{if}(b)\ \{C\}\ \texttt{else}\ \{C\} \mid x := e \mid \texttt{skip} \mid \texttt{custom}\ K$$

# CoreIVL: A Core Language for SL-based IVLs

Gillian VeriFast VIPER

Adds resources

Removes resources

$$C ::= \texttt{inhale}\ A \mid \texttt{exhale}\ A \mid \texttt{havoc}\ x \mid C;C \mid \texttt{if}(b)\ \{C\}\ \texttt{else}\ \{C\} \mid x := e \mid \texttt{skip} \mid \texttt{custom}\ K$$

# CoreIVL: A Core Language for SL-based IVLs

Gillian VeriFast VIPER

Adds resources

Removes resources

$$C ::= \texttt{inhale}\ A \mid \texttt{exhale}\ A \mid \texttt{havoc}\ x \mid C;C \mid \texttt{if}(b)\ \{C\}\ \texttt{else}\ \{C\} \mid x := e \mid \texttt{skip} \mid \texttt{custom}\ K$$

Non deterministic assignment

# CoreIVL: A Core Language for SL-based IVLs

Gillian VeriFast VIPER

Adds resources

Removes resources

Custom statements for IVL, e.g. field assignment

$$C ::= \texttt{inhale}\ A \mid \texttt{exhale}\ A \mid \texttt{havoc}\ x \mid C;C \mid \texttt{if}(b)\ \{C\}\ \texttt{else}\ \{C\} \mid x := e \mid \texttt{skip} \mid \texttt{custom}\ K$$

Non deterministic assignment

# CoreIVL: A Core Language for SL-based IVLs


assertions based on *separation logic*


$$C ::= \texttt{inhale}\ A \mid \texttt{exhale}\ A \mid \texttt{havoc}\ x \mid C;C \mid \texttt{if}(b)\ \{C\}\ \texttt{else}\ \{C\} \mid x := e \mid \texttt{skip} \mid \texttt{custom}\ K$$

# CoreIVL: A Core Language for SL-based IVLs


assertions based on *separation logic*


assertions based on *implicit dynamic frames*

$$C ::= \texttt{inhale}\ A \mid \texttt{exhale}\ A \mid \texttt{havoc}\ x \mid C;C \mid \texttt{if}(b)\ \{C\}\ \texttt{else}\ \{C\} \mid x := e \mid \texttt{skip} \mid \texttt{custom}\ K$$

# CoreIVL: A Core Language for SL-based IVLs

Gillian VeriFast

assertions based on *separation logic*

VIPER

assertions based on *implicit dynamic frames*

$$C ::= \texttt{inhale}\ A \mid \texttt{exhale}\ A \mid \texttt{havoc}\ x \mid C;C \mid \texttt{if}(b)\ \{C\}\ \texttt{else}\ \{C\} \mid x := e \mid \texttt{skip} \mid \texttt{custom}\ K$$

Assertions parametric over novel **generalized separation algebra**
(capturing both *separation logic* and *implicit dynamic frames*)

# CoreIVL: A Core Language for SL-based IVLs


assertions based on *separation logic*

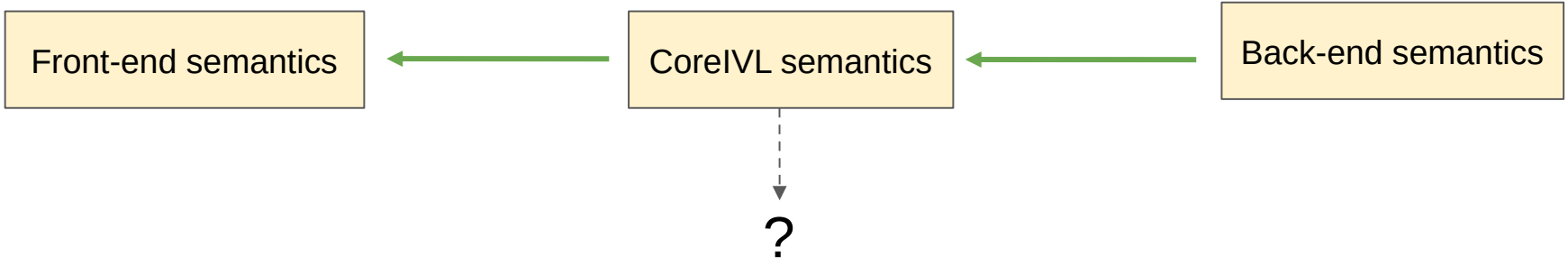

assertions based on *implicit dynamic frames*

$$C ::= \texttt{inhale}\ A \mid \texttt{exhale}\ A \mid \texttt{havoc}\ x \mid C;C \mid \texttt{if}(b)\ \{C\}\ \texttt{else}\ \{C\} \mid x := e \mid \texttt{skip} \mid \texttt{custom}\ K$$

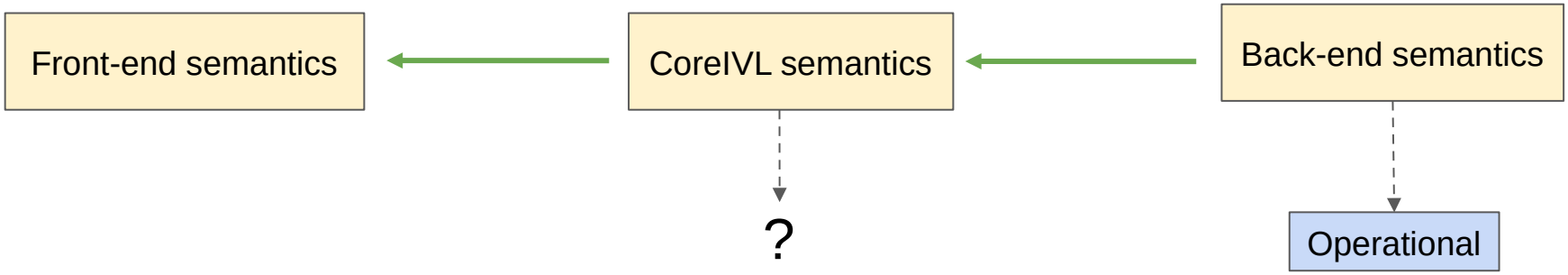
Assertions parametric over novel **generalized separation algebra**
(capturing both *separation logic* and *implicit dynamic frames*)

What kind of semantics for CoreIVL?

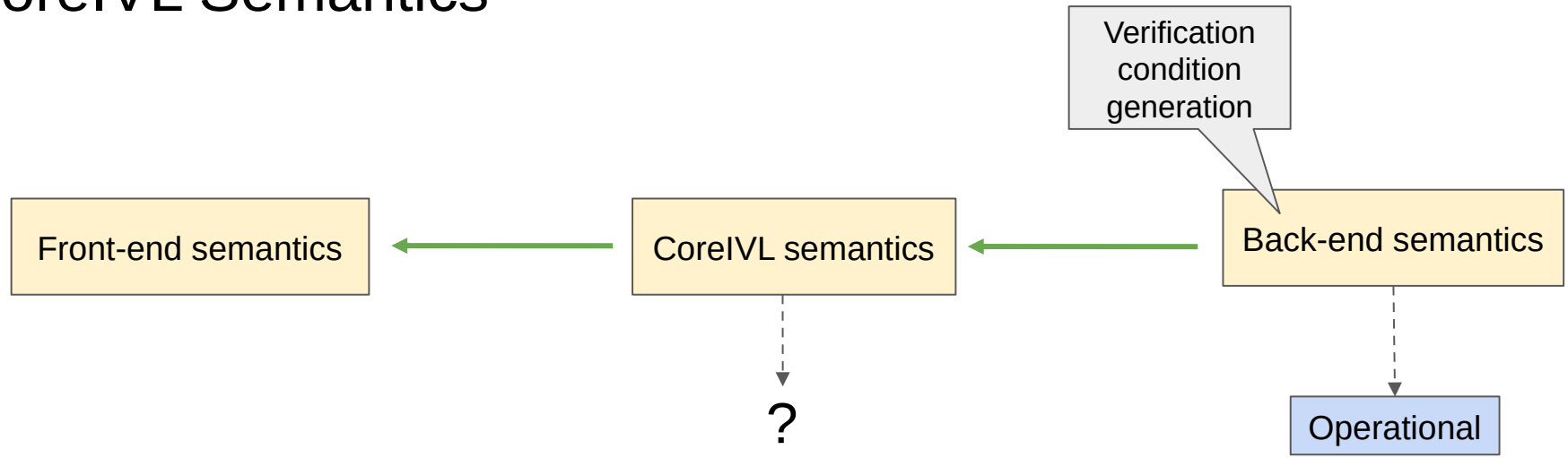
# CoreIVL Semantics



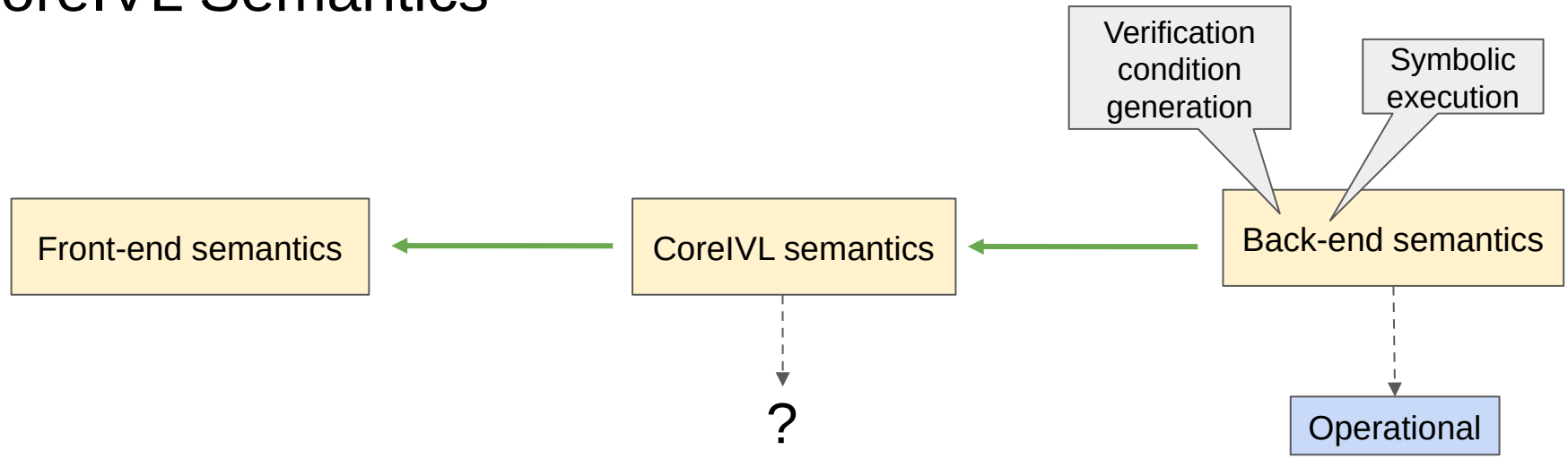
# CoreIVL Semantics



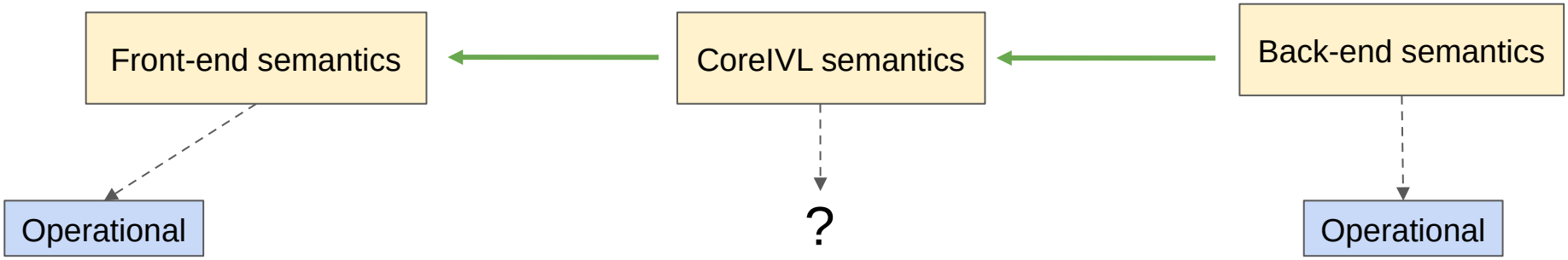
# CoreIVL Semantics



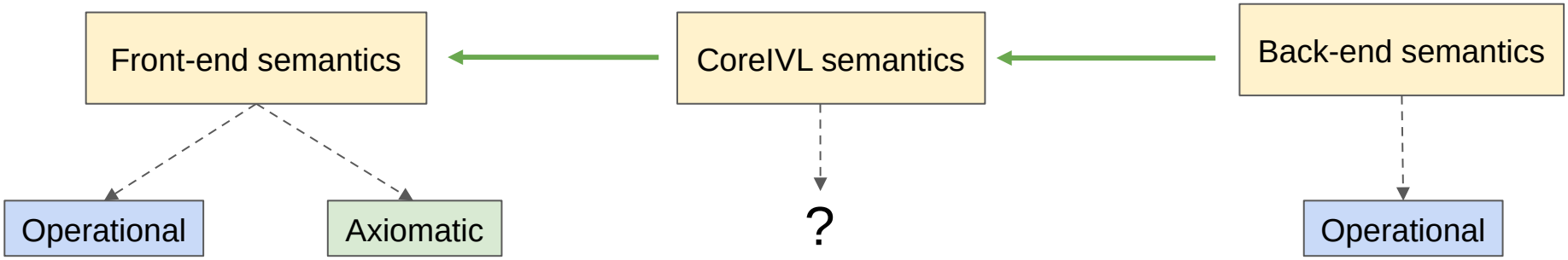
# CoreIVL Semantics



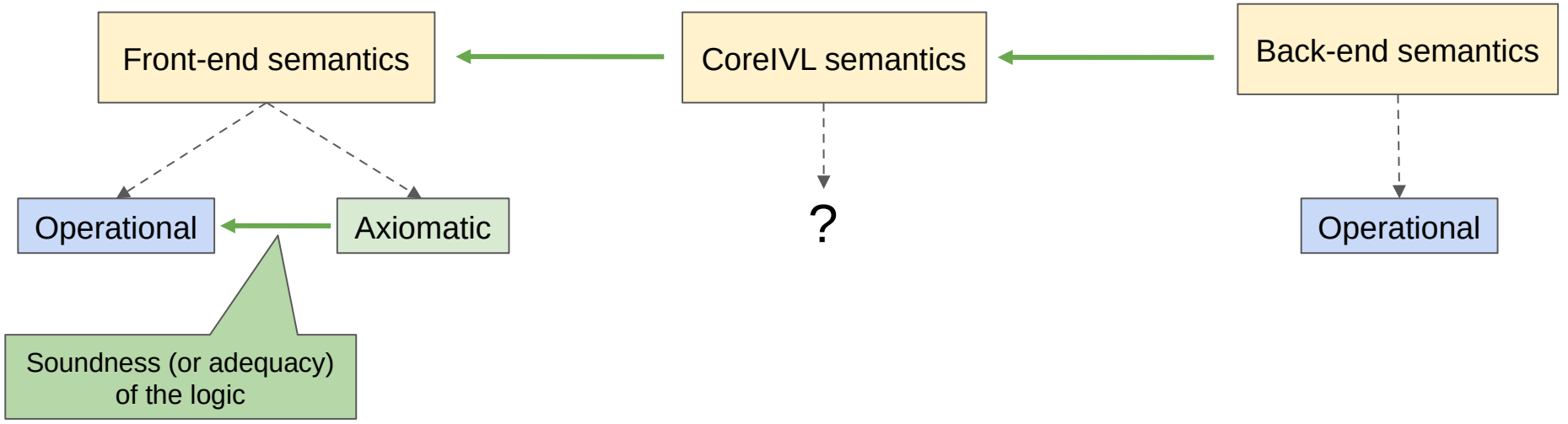
# CoreIVL Semantics



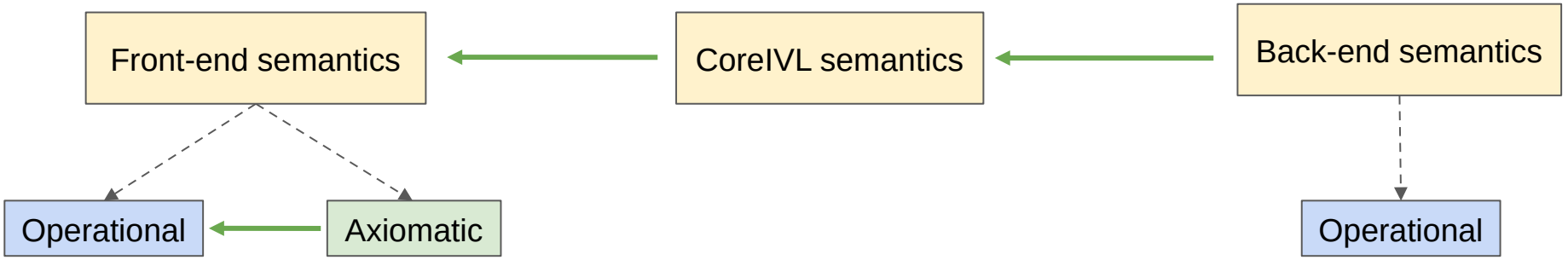
# CoreIVL Semantics



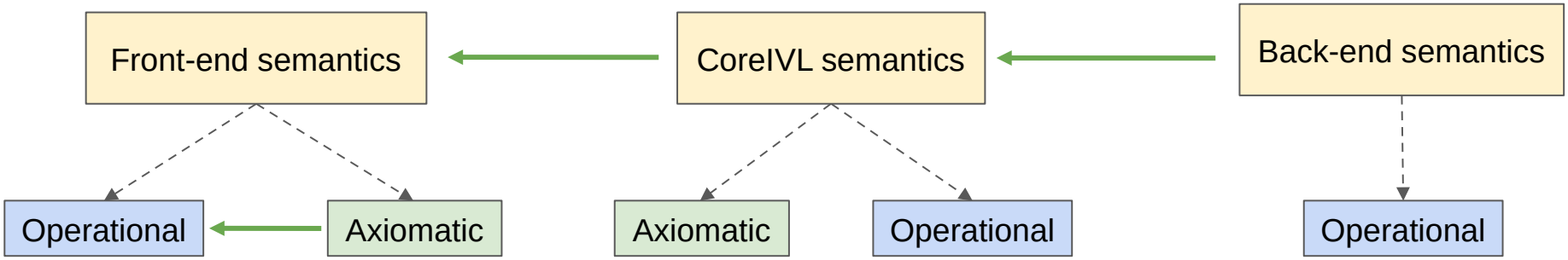
# CoreIVL Semantics



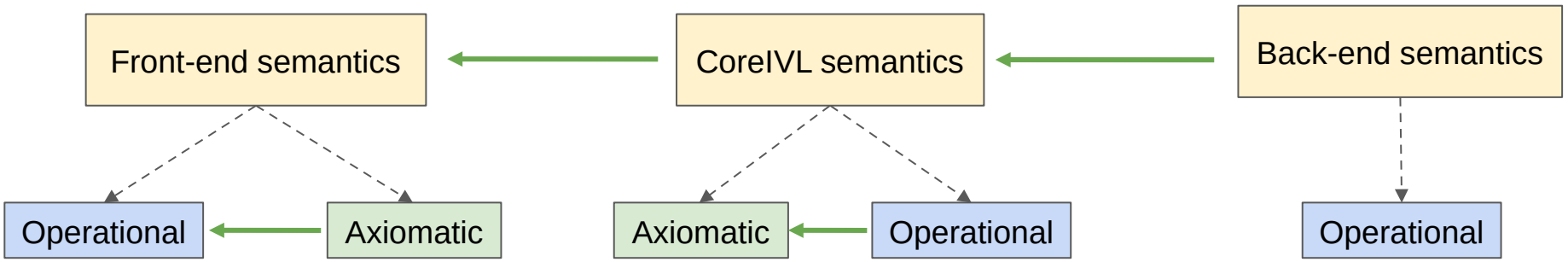
# Key Idea: Two CoreIVL Semantics



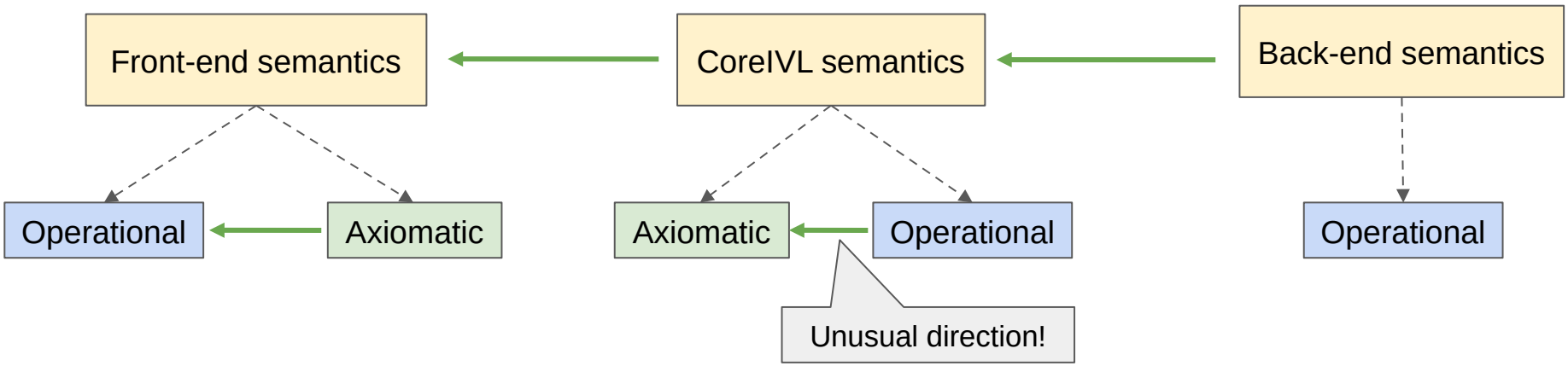
# Key Idea: Two CoreIVL Semantics



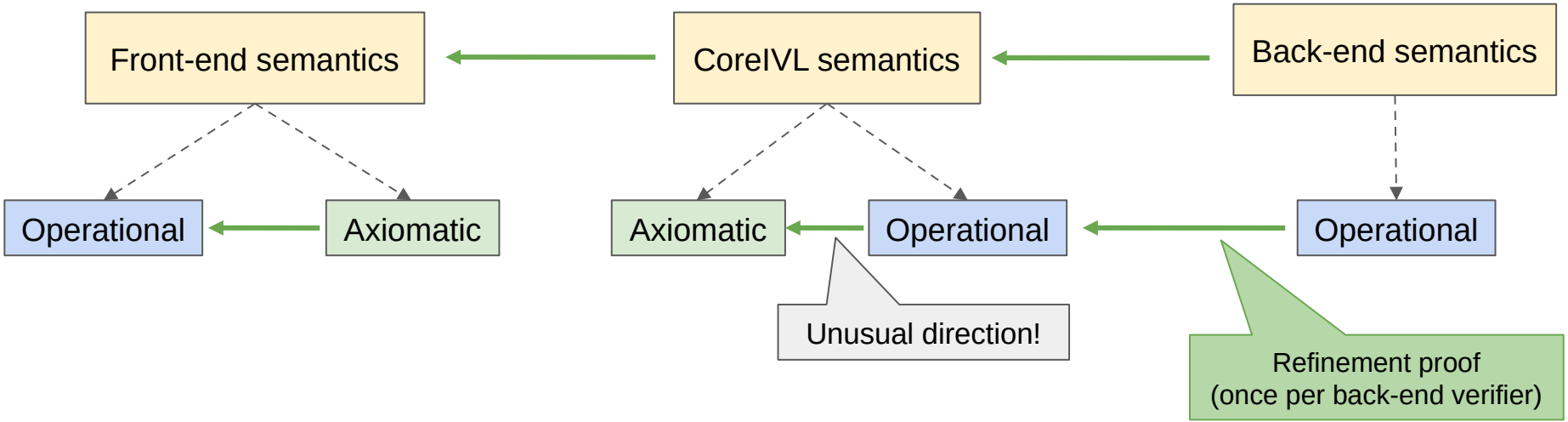
# Key Idea: Two CoreIVL Semantics



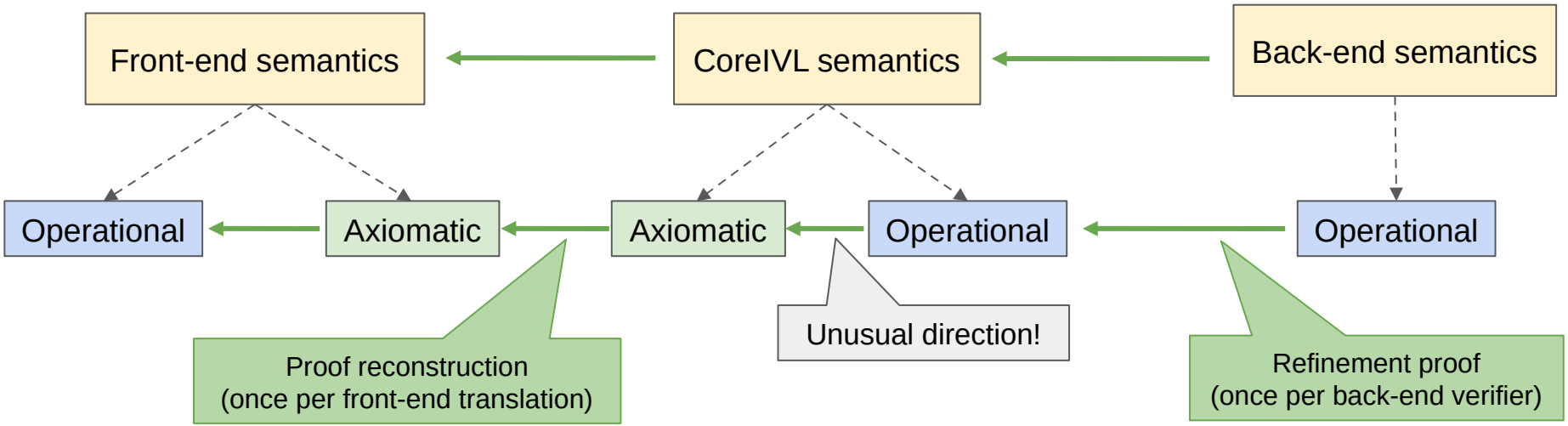
# Key Idea: Two CoreIVL Semantics



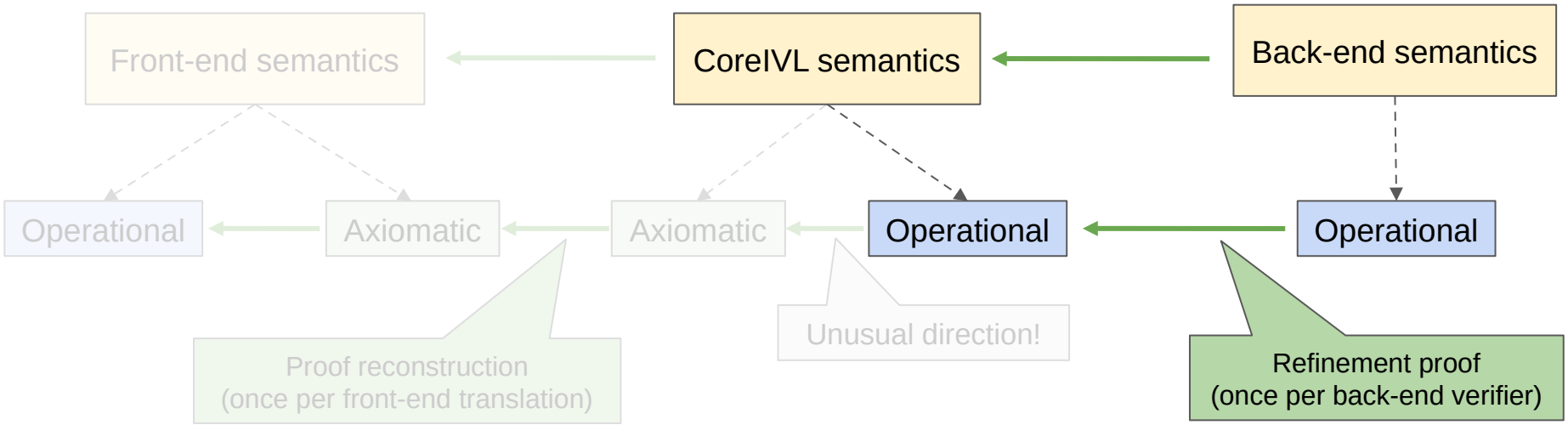
# Key Idea: Two CoreIVL Semantics



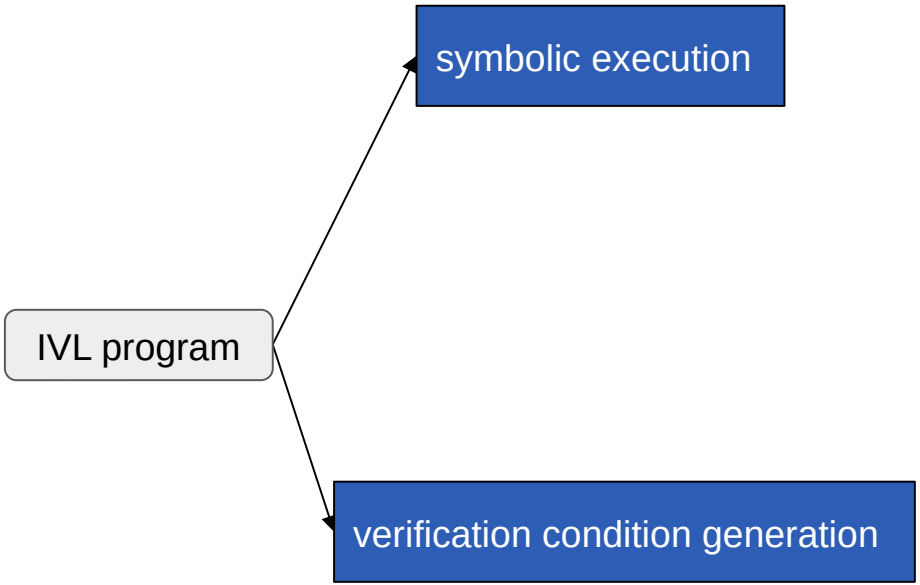
# Key Idea: Two CoreIVL Semantics



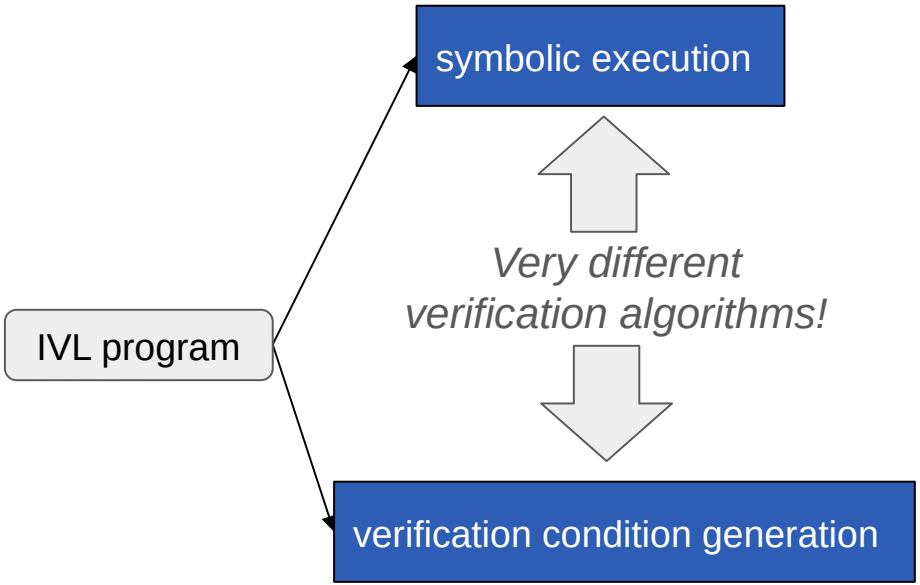
# Key Idea: Two CoreIVL Semantics



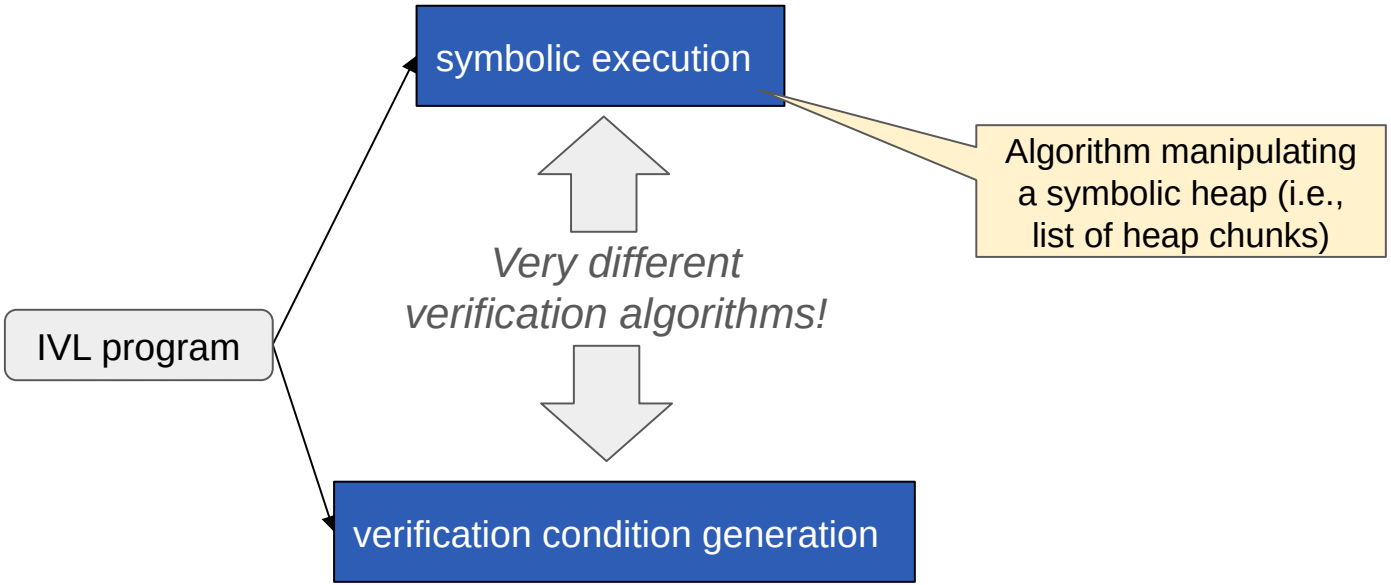
# Different Verification Algorithms



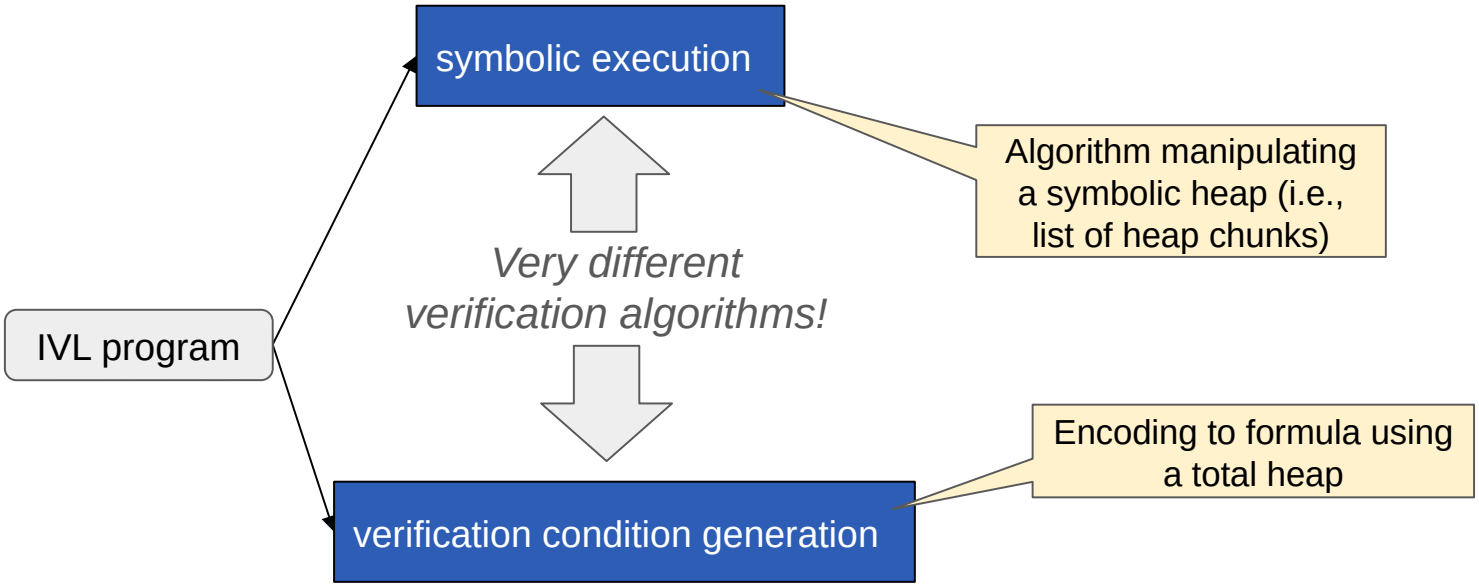
# Different Verification Algorithms



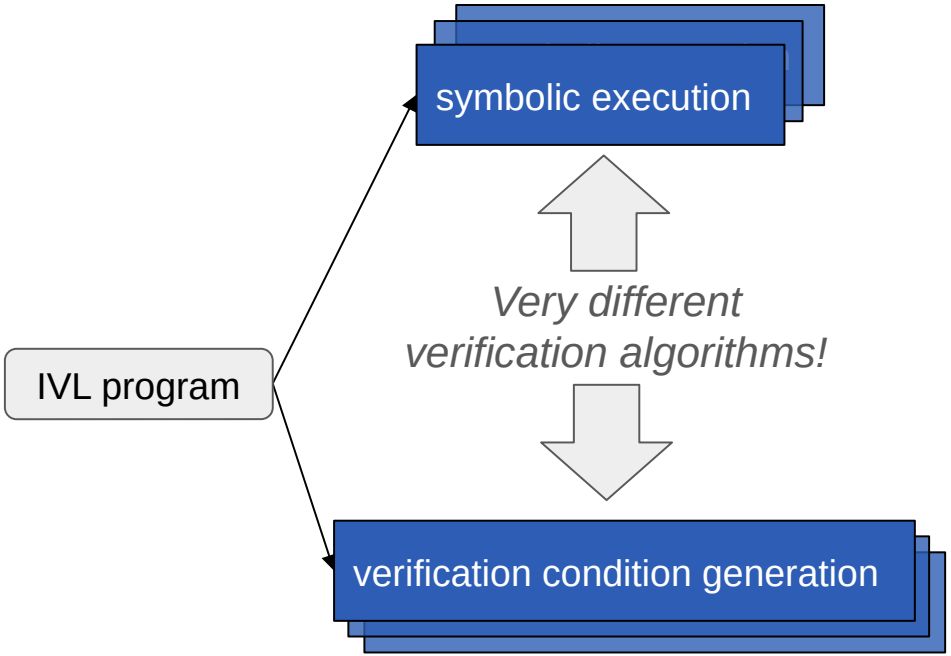
# Different Verification Algorithms



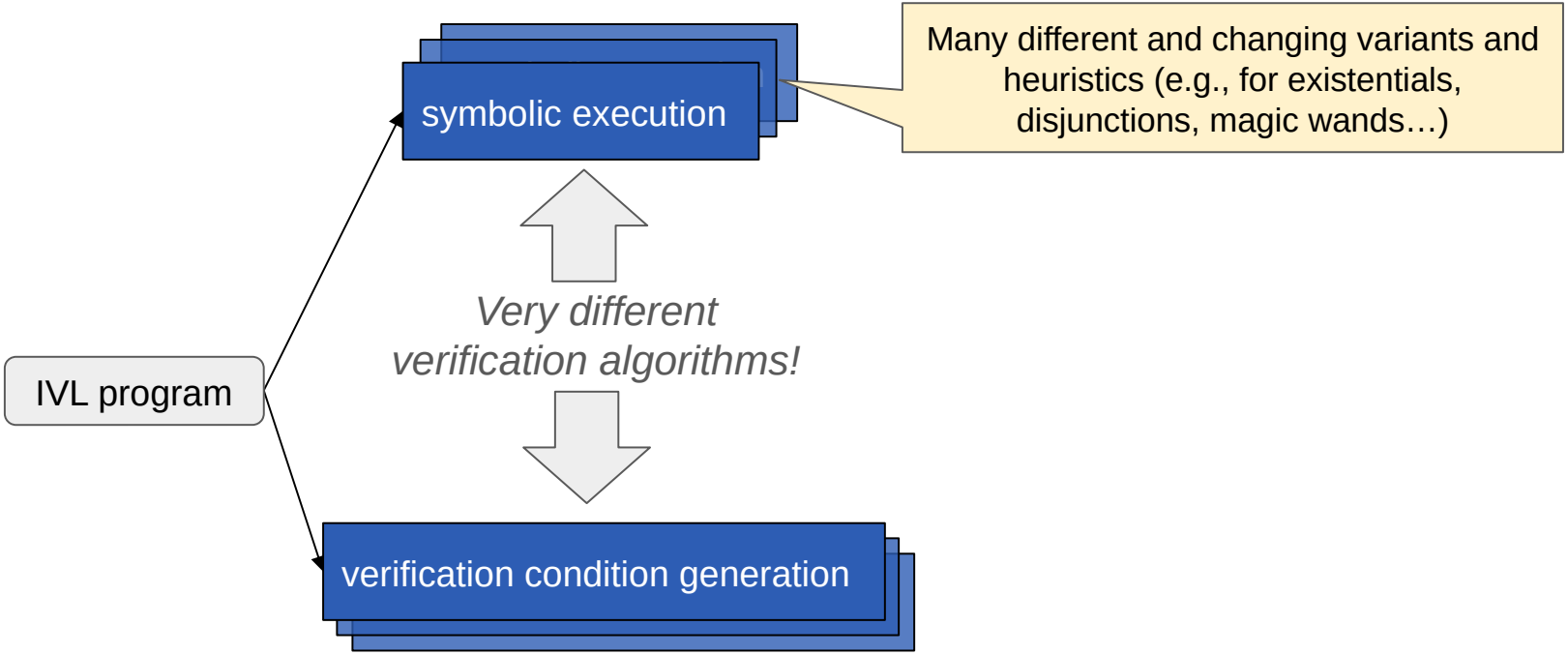
# Different Verification Algorithms



# Different Verification Algorithms



# Different Verification Algorithms



# Operational Semantics of Exhale

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

?

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)

?

# Operational Semantics of Exhale

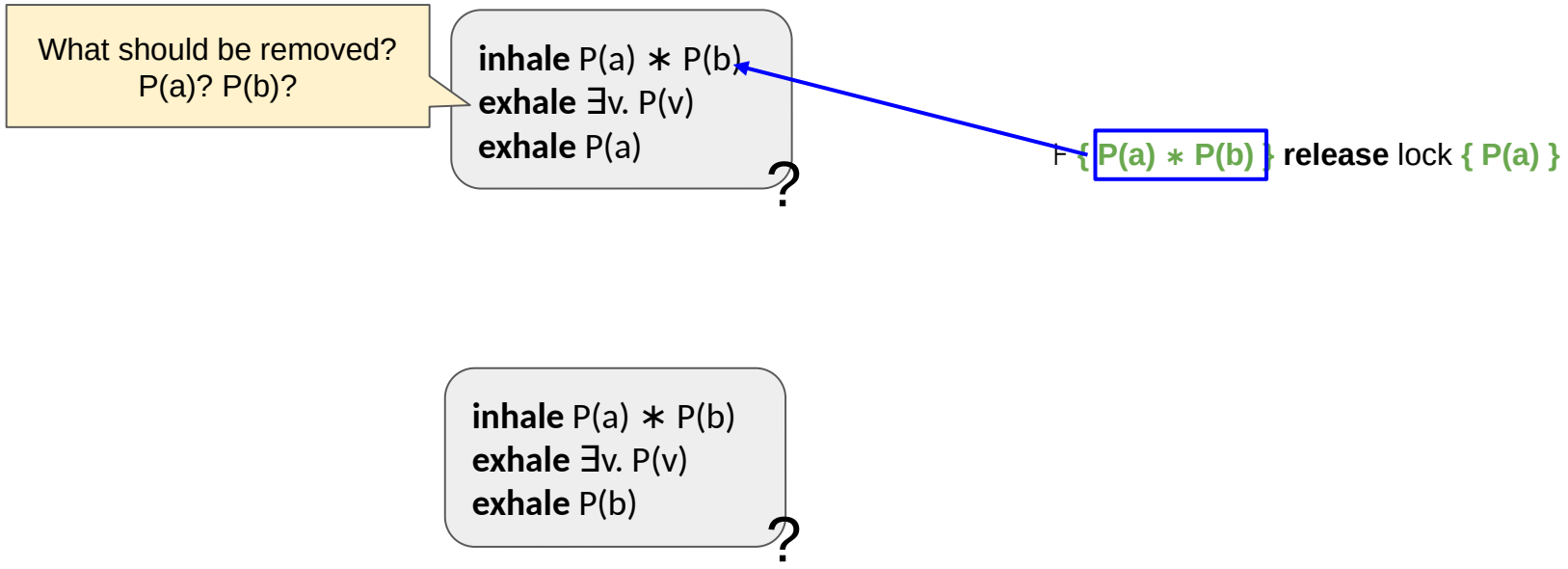
What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)
?

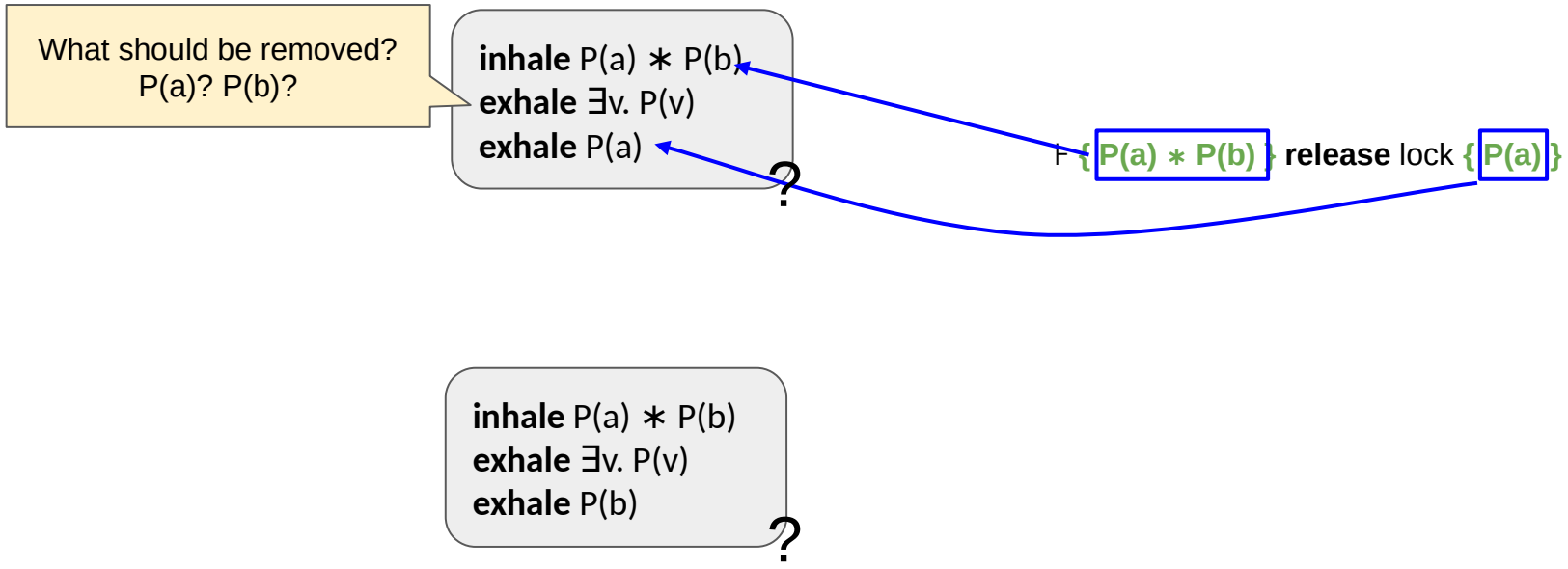
⊢ { P(a) ∗ P(b) } **release** lock { P(a) }

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)
?

# Operational Semantics of Exhale



# Operational Semantics of Exhale



# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)
?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)
?

⊢ **{ ∃v. P(v) }** **release** lock **{ ⊤ }**

⊢ **{ P(a) ∗ P(b) }** **release** lock **{ P(a) }**

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)
?

⊢ { ∃v. P(v) } **release** lock { ⊤ }

⊢ { P(a) ∗ P(b) } **release** lock { P(a) }

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)
?

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)

?

$$\frac{\vdash \{\ \exists v.\ P(v)\ \}\ \textbf{release}\ \text{lock}\ \{\ \top\ \}}{\vdash \{\ P(b)\ \}\ \textbf{release}\ \text{lock}\ \{\ \top\ \}}\ (\text{Cons})$$

$$\frac{\vdash \{\ P(b)\ \}\ \textbf{release}\ \text{lock}\ \{\ \top\ \}}{\vdash \{\ P(a) * P(b)\ \}\ \textbf{release}\ \text{lock}\ \{\ P(a)\ \}}\ (\text{Frame})$$

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)
?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)
?

$$\frac{\vdash \{\ \exists v.\ P(v)\ \}\ \textbf{release}\ \text{lock}\ \{\ \top\ \}}{\vdash \{\ P(b)\ \}\ \textbf{release}\ \text{lock}\ \{\ \top\ \}}\ (\text{Cons})$$

$$\frac{\vdash \{\ P(b)\ \}\ \textbf{release}\ \text{lock}\ \{\ \top\ \}}{\vdash \{\ P(a) * P(b)\ \}\ \textbf{release}\ \text{lock}\ \{\ P(a)\ \}}\ (\text{Frame})$$

with frame *P(a)*

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)

?

$$\frac{\vdash \{\ \exists v.\ P(v)\ \}\ \textbf{release}\ \text{lock}\ \{\ \top\ \}}{\vdash \{\ P(b)\ \}\ \textbf{release}\ \text{lock}\ \{\ \top\ \}}\ (\text{Cons})$$

$$\frac{\vdash \{\ P(b)\ \}\ \textbf{release}\ \text{lock}\ \{\ \top\ \}}{\vdash \{\ P(a) * P(b)\ \}\ \textbf{release}\ \text{lock}\ \{\ P(a)\ \}}\ (\text{Frame})$$

with frame *P(a)*

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

$$\frac{\vdash \{\ \exists v.\ P(v)\ \}\ \textbf{release lock}\ \{\ \top\ \}}{\vdash \{\ P(b)\ \}\ \textbf{release lock}\ \{\ \top\ \}}\ (\text{Cons})$$
$$\frac{\vdash \{\ P(b)\ \}\ \textbf{release lock}\ \{\ \top\ \}}{\vdash \{\ P(a) * P(b)\ \}\ \textbf{release lock}\ \{\ P(a)\ \}}\ (\text{Frame})$$

with frame *P(a)*

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)

?

$$\frac{\vdash \{\ \exists v.\ P(v)\ \}\ \textbf{release lock}\ \{\ \top\ \}}{\vdash \{\ P(a)\ \}\ \textbf{release lock}\ \{\ \top\ \}}\ (\text{Cons})$$
$$\frac{\vdash \{\ P(a)\ \}\ \textbf{release lock}\ \{\ \top\ \}}{\vdash \{\ P(a) * P(b)\ \}\ \textbf{release lock}\ \{\ P(b)\ \}}\ (\text{Frame})$$

with frame *P(b)*

# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

$$\frac{\vdash \{\ \exists v.\ P(v)\ \}\ \textbf{release lock}\ \{\ \top\ \}}{\vdash \{\ P(b)\ \}\ \textbf{release lock}\ \{\ \top\ \}}\ (\text{Cons})$$
$$\frac{\vdash \{\ P(b)\ \}\ \textbf{release lock}\ \{\ \top\ \}}{\vdash \{\ P(a) * P(b)\ \}\ \textbf{release lock}\ \{\ P(a)\ \}}\ (\text{Frame})$$

with frame *P(a)*

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)

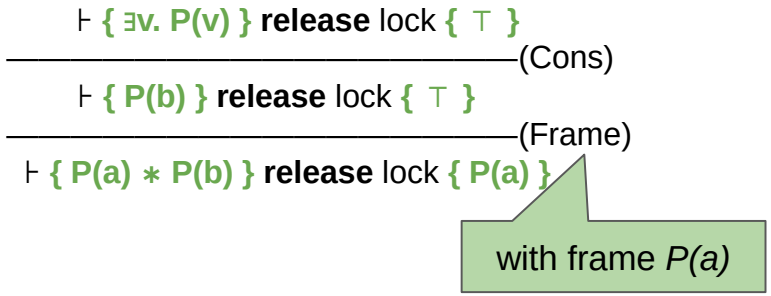
$$\frac{\vdash \{\ \exists v.\ P(v)\ \}\ \textbf{release lock}\ \{\ \top\ \}}{\vdash \{\ P(a)\ \}\ \textbf{release lock}\ \{\ \top\ \}}\ (\text{Cons})$$
$$\frac{\vdash \{\ P(a)\ \}\ \textbf{release lock}\ \{\ \top\ \}}{\vdash \{\ P(a) * P(b)\ \}\ \textbf{release lock}\ \{\ P(b)\ \}}\ (\text{Frame})$$

with frame *P(b)*

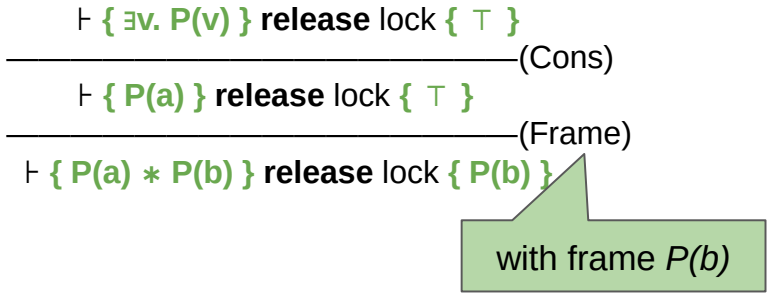
# Operational Semantics of Exhale

What should be removed?
P(a)? P(b)?

**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(a)

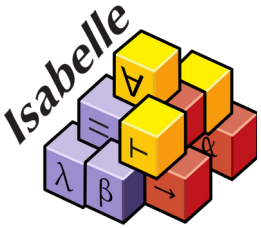



**inhale** P(a) ∗ P(b)
**exhale** ∃v. P(v)
**exhale** P(b)




**Angelic non-determinism** in the operational semantics captures the different behaviors (e.g., different choices of frame) of different back-end verifiers.

# Contributions





# Contributions




Generic CoreIVL framework

# Contributions



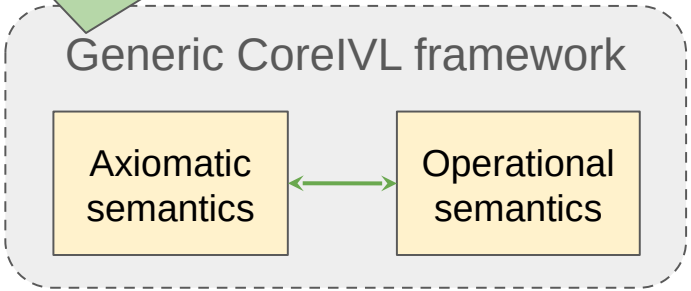

Parametric over a generalized separation algebra
(for *separation logic* and *implicit dynamic frames*)

Generic CoreIVL framework

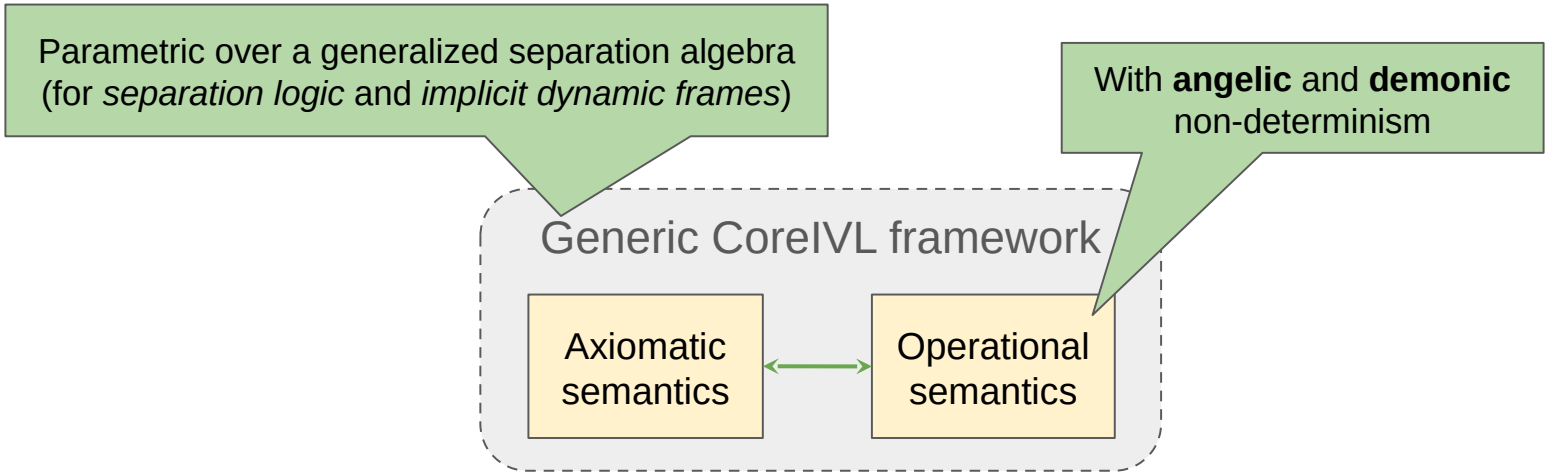
# Contributions



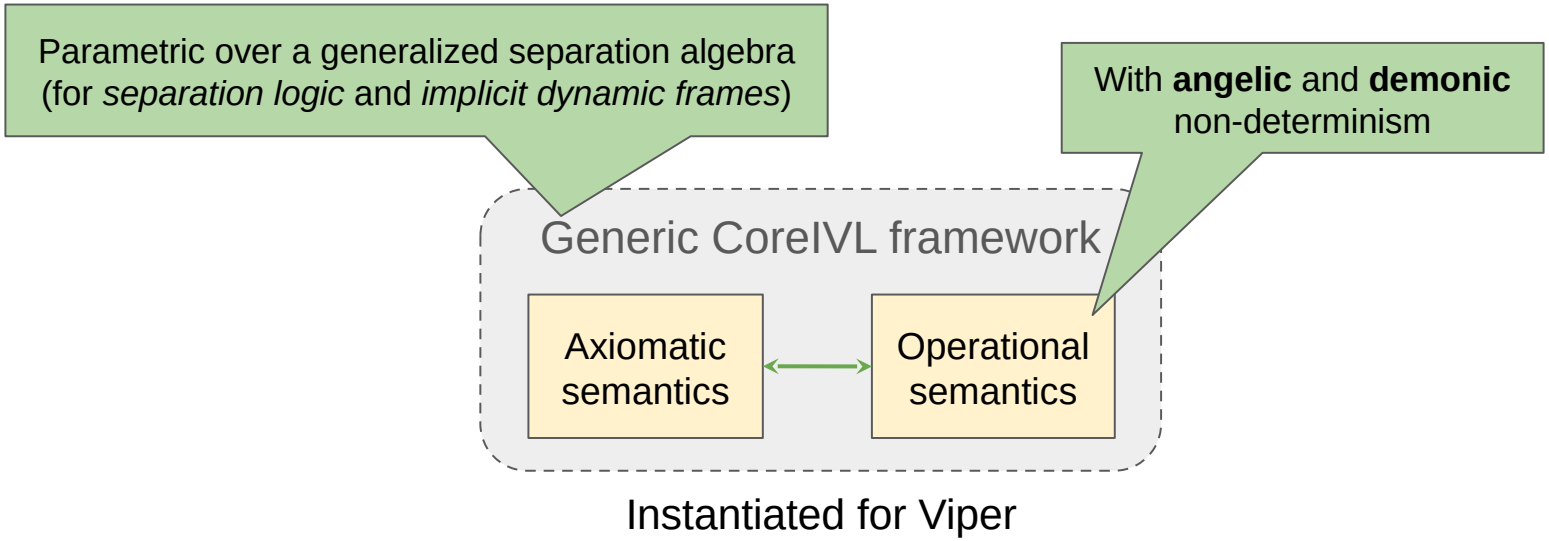

Parametric over a generalized separation algebra
(for *separation logic* and *implicit dynamic frames*)



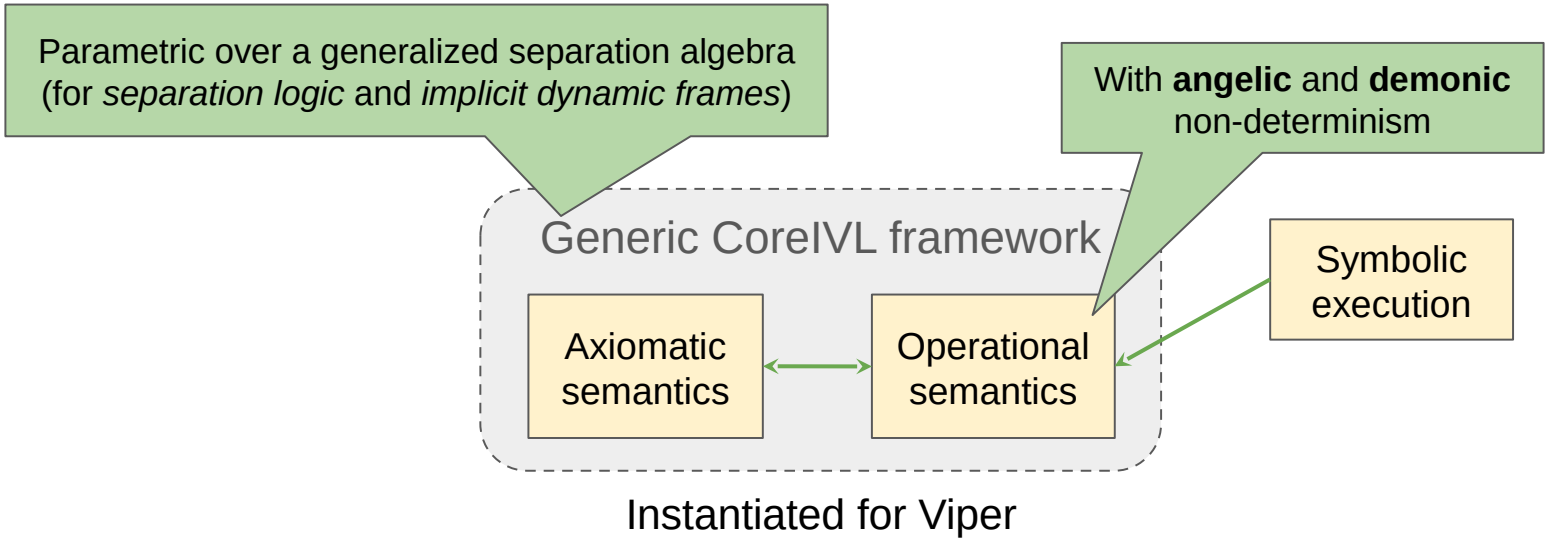
# Contributions



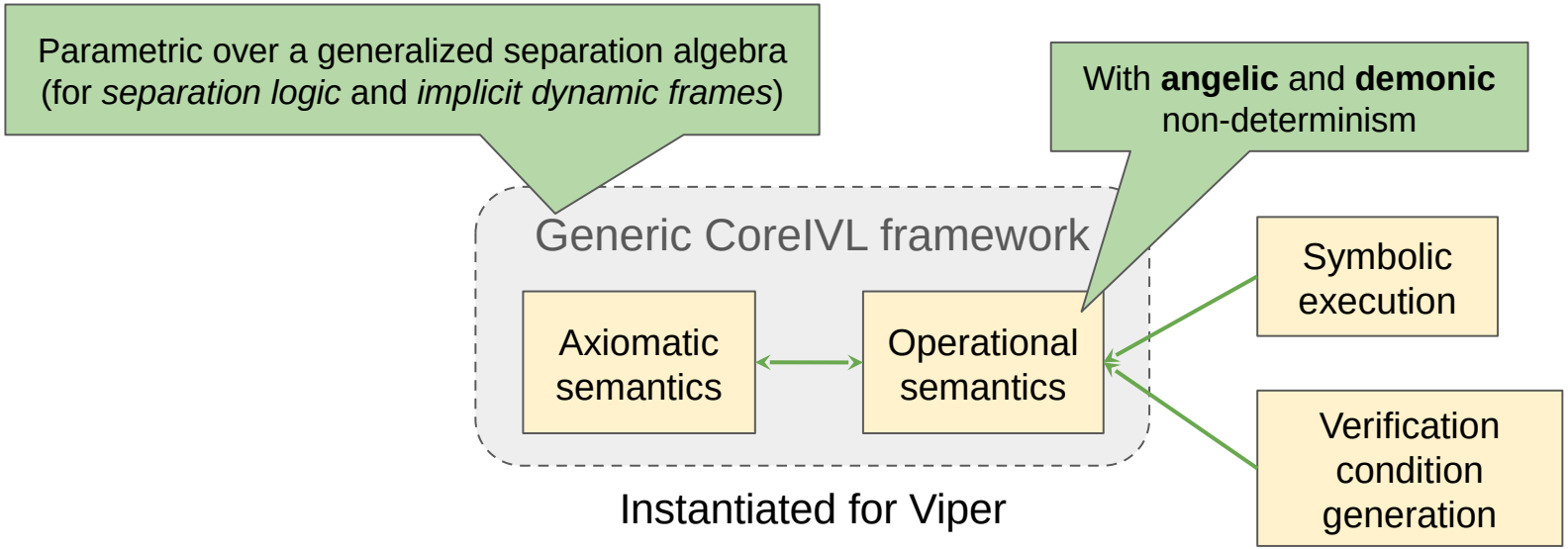
# Contributions



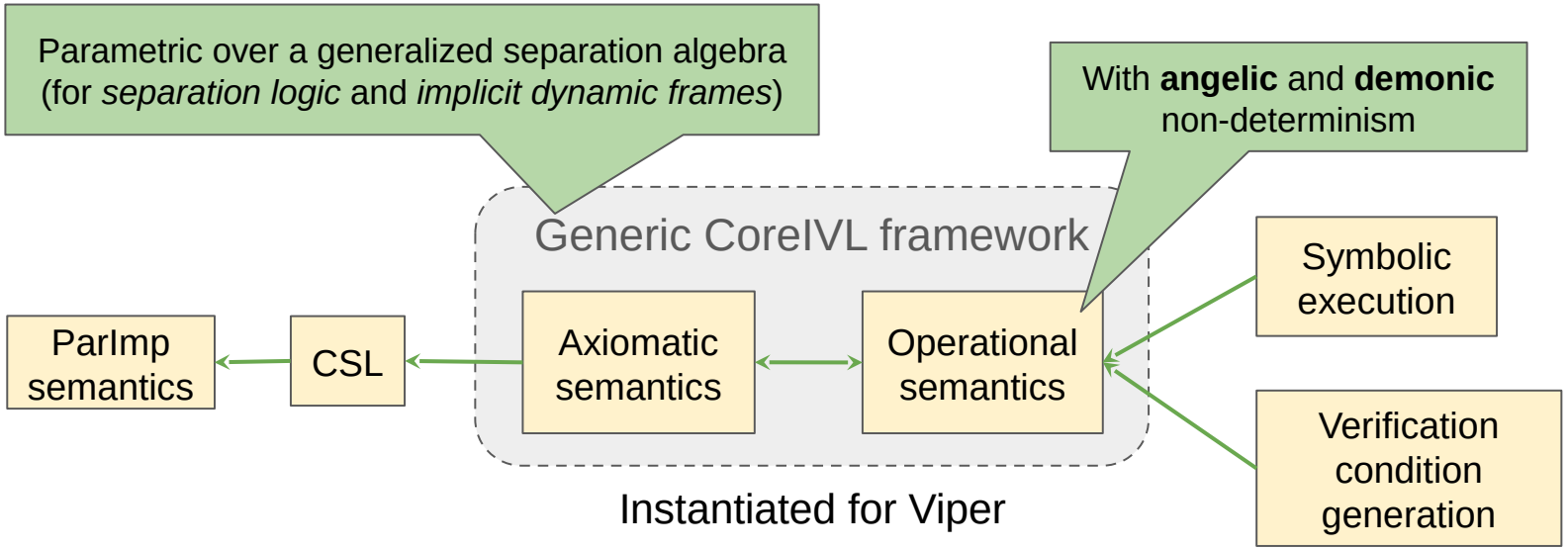
# Contributions



# Contributions



# Contributions



# Contributions