

Hyper Hoare Logic

(Dis-)Proving Program Hyperproperties

Thibault Dardinier, Peter Müller

ETH zürich

Hyperproperties

Hyperproperty $\triangleq$ property of a **set** of executions

	Type	Type of negation
Non-interference	$\forall\forall$	$\exists\exists$
Determinism	$\forall\forall$	$\exists\exists$
Monotonicity	$\forall\forall$	$\exists\exists$
Transitivity	$\forall\forall\forall$	$\exists\exists\exists$
Functional correctness	$\forall$	$\exists$
Reachability	$\exists$	$\forall$
Generalized non-interference	$\forall\forall\exists$	$\exists\exists\forall$
Existence of a minimum	$\exists\forall$	$\forall\exists$

Hyperproperties

Hyperproperty $\triangleq$ property of a **set** of executions

For any two executions, if they have the same public inputs, they must have the same public outputs

		Type of negation
Non-interference	$\forall\forall$	$\exists\exists$
Determinism	$\forall\forall$	$\exists\exists$
Monotonicity	$\forall\forall$	$\exists\exists$
Transitivity	$\forall\forall\forall$	$\exists\exists\exists$
Functional correctness	$\forall$	$\exists$
Reachability	$\exists$	$\forall$
Generalized non-interference	$\forall\forall\exists$	$\exists\exists\forall$
Existence of a minimum	$\forall\exists$	$\exists\forall$

Hyperproperties

Hyperproperty $\triangleq$ property of a **set** of executions

	Type	
Non-interference	$\forall\forall$	$\exists\exists$
Determinism	$\forall\forall$	$\exists\exists$
Monotonicity	$\forall\forall$	$\exists\exists$
Transitivity	$\forall\forall\forall$	$\exists\exists\exists$
Functional correctness	$\forall$	$\exists$
Reachability	$\exists$	$\forall$
Generalized non-interference	$\forall\forall\exists$	$\exists\exists\forall$
Existence of a minimum	$\forall\exists$	$\exists\forall$

2-safety hyperproperty

Hyperproperties

Hyperproperty $\triangleq$ property of a **set** of executions

	Type	Type of negation
Non-interference	$\forall\forall$	$\exists\exists$
Determinism	$\forall\forall$	$\exists\exists$
Monotonicity	$\forall\forall$	$\exists\exists$
Transitivity	$\forall\forall\forall$	$\exists\exists\exists$
Functional correctness	Intuitively, " $\forall x. \forall y. x \geq y \Rightarrow f(x) \geq f(y)$ "	
Reachability	$\exists$	$\forall$
Generalized non-interference	$\forall\forall\exists$	$\exists\exists\forall$
Existence of a minimum	$\forall\exists$	$\exists\forall$

Hyperproperties

Hyperproperty $\triangleq$ property of a **set** of executions

	Type	Type of negation
Non-interference	$\forall\forall$	$\exists\exists$
Determinism	$\forall\forall$	$\exists\exists$
Monotonicity	$\forall\forall$	$\exists\exists$
Transitivity	$\forall\forall\forall$	$\exists\exists\exists$
Functional correctness	$\forall$	$\exists$
Reachability	$\exists$	$\forall$
Generalized non-interference	$\forall\forall\exists$	$\exists\exists\forall$
Existence of a minimum	$\forall\exists$	$\exists\forall$

Intuitively, “ $\forall x. \forall y. \forall z. f(x, y) \wedge f(y, z) \Rightarrow f(x, z)$ ”

Hyperproperties

Hyperproperty $\triangleq$ property of a **set** of executions

Intuitively, “ $\exists x. \exists y. x \geq y \wedge f(x) < f(y)$ ”

	Type	Type
Non-interference	$\forall\forall$	$\exists\exists$
Determinism	$\forall\forall$	$\exists\exists$
Monotonicity	$\forall\forall$	$\exists\exists$
Transitivity	$\forall\forall\forall$	$\exists\exists\exists$
Functional correctness	$\forall$	$\exists$
Reachability	$\exists$	$\forall$
Generalized non-interference	$\forall\forall\exists$	$\exists\exists\forall$
Existence of a minimum	$\forall\exists$	$\exists\forall$

Hyperproperties

Hyperproperty $\triangleq$ property of a set of executions

Goal

Develop the first program logic that can handle all these different types of hyperproperties.

Reachability	$\exists$	$\forall$
Generalized non-interference	$\forall\forall\exists$	$\exists\exists\forall$
Existence of a minimum	$\exists\forall$	$\forall\exists$

Main contribution

Hyper Hoare Logic, a program logic that allows to (dis-)prove arbitrary program hyperproperties, including $\forall^*\exists^*$ - and $\exists^*\forall^*$ -hyperproperties.

Contributions

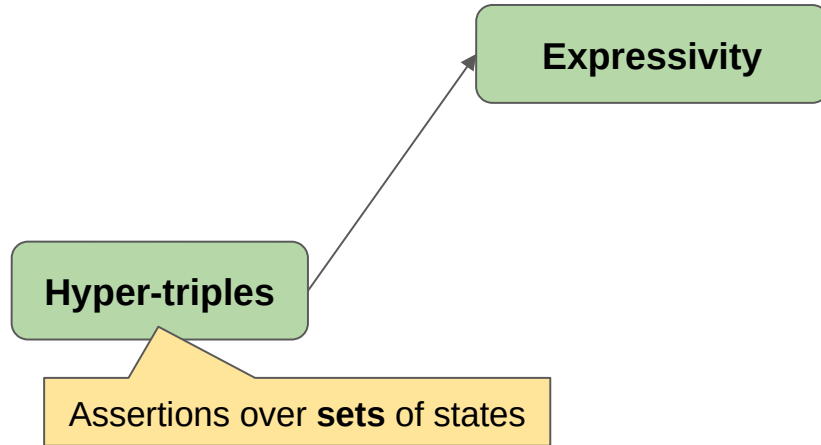
Hyper-triples

Contributions

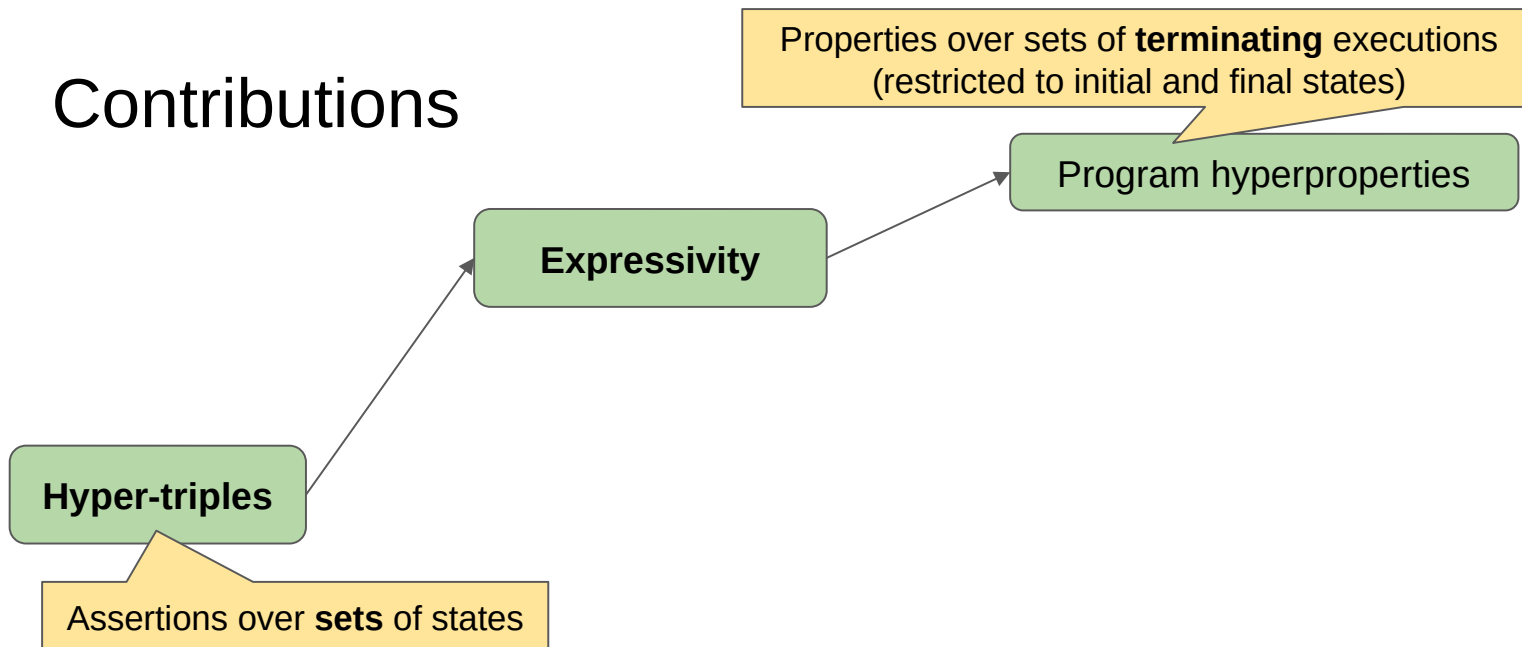
Hyper-triples

Assertions over **sets** of states

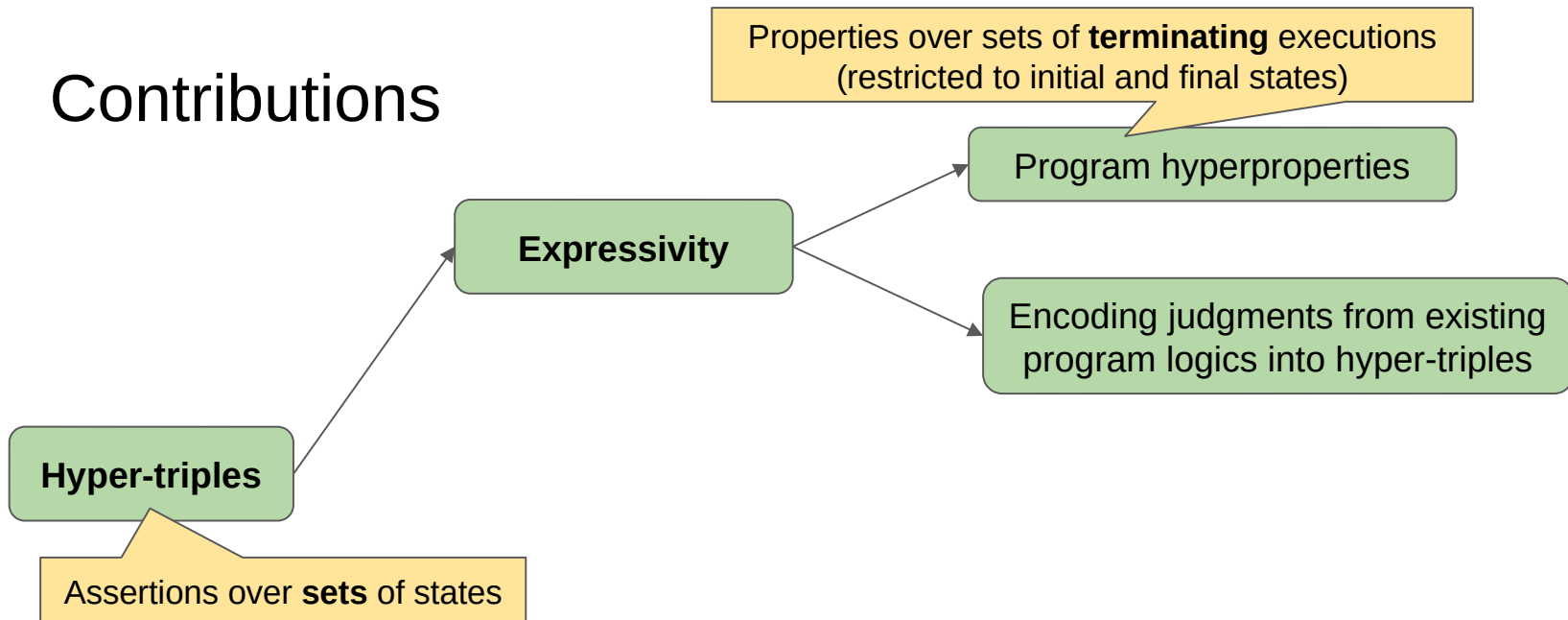
Contributions



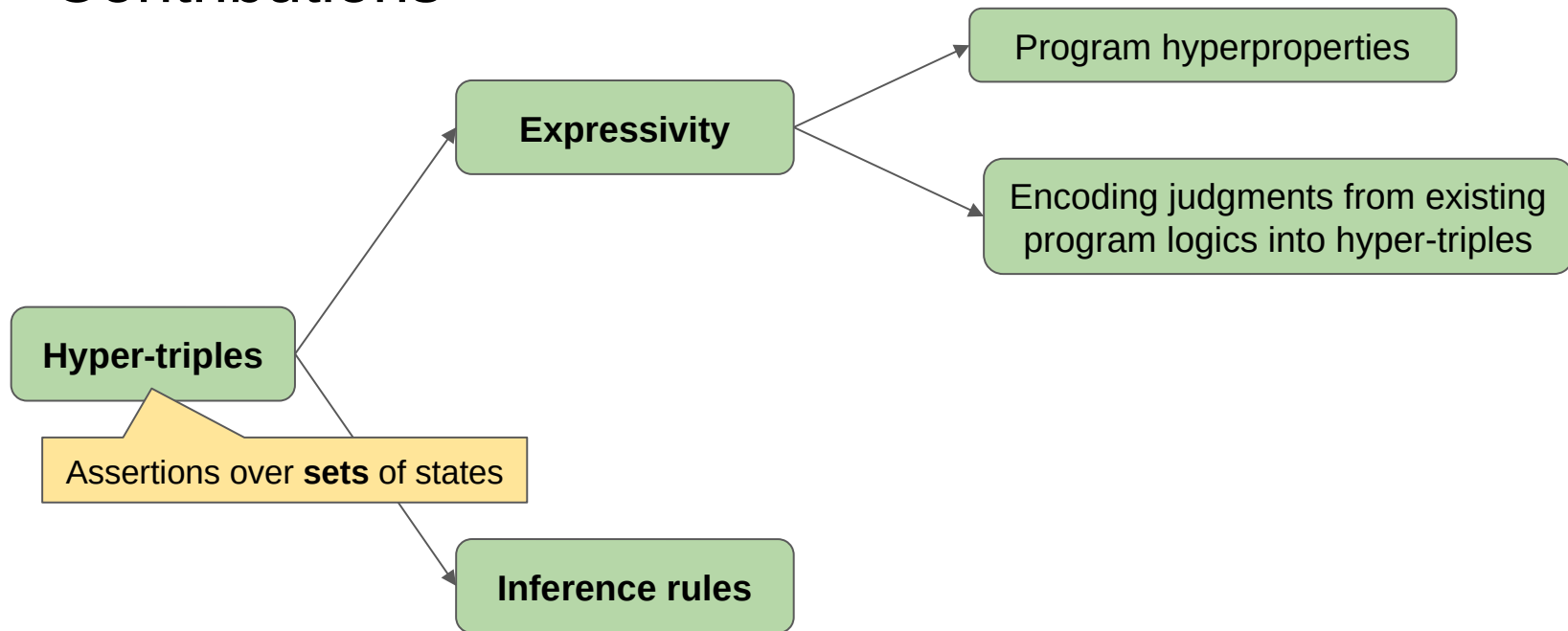
Contributions



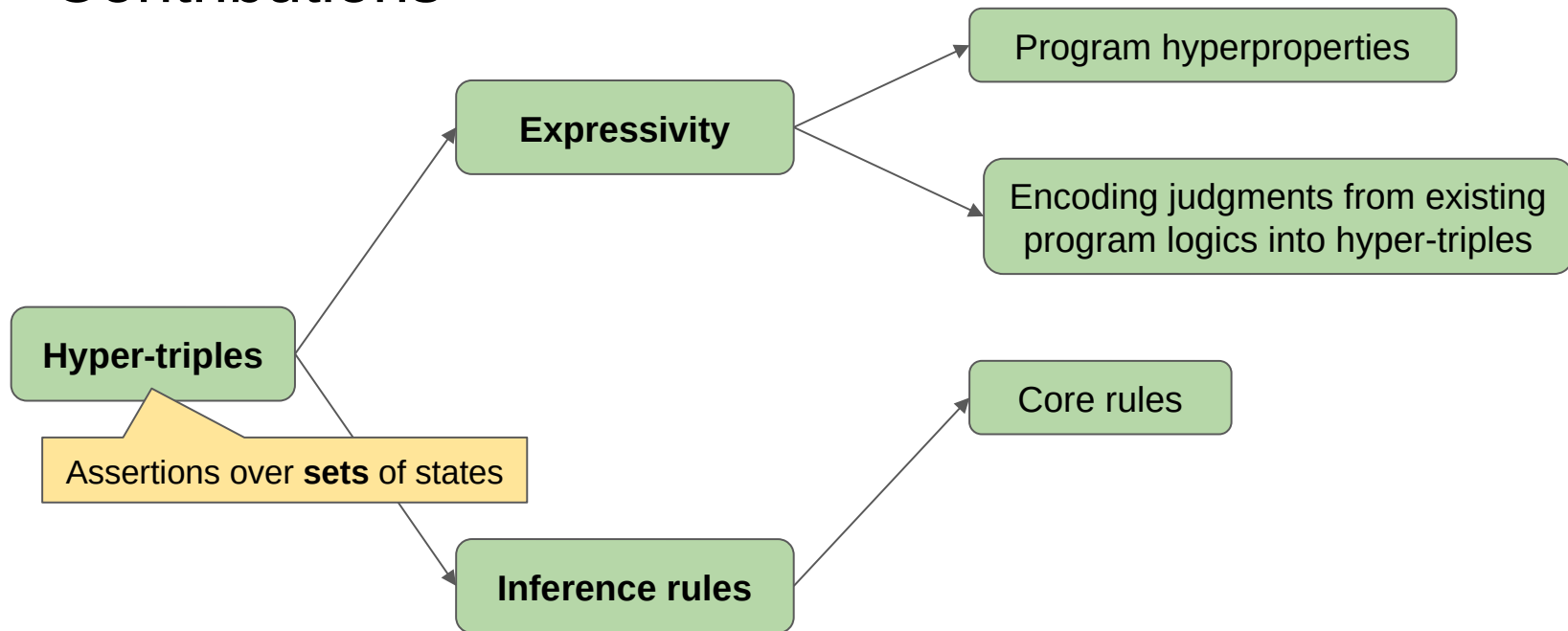
Contributions



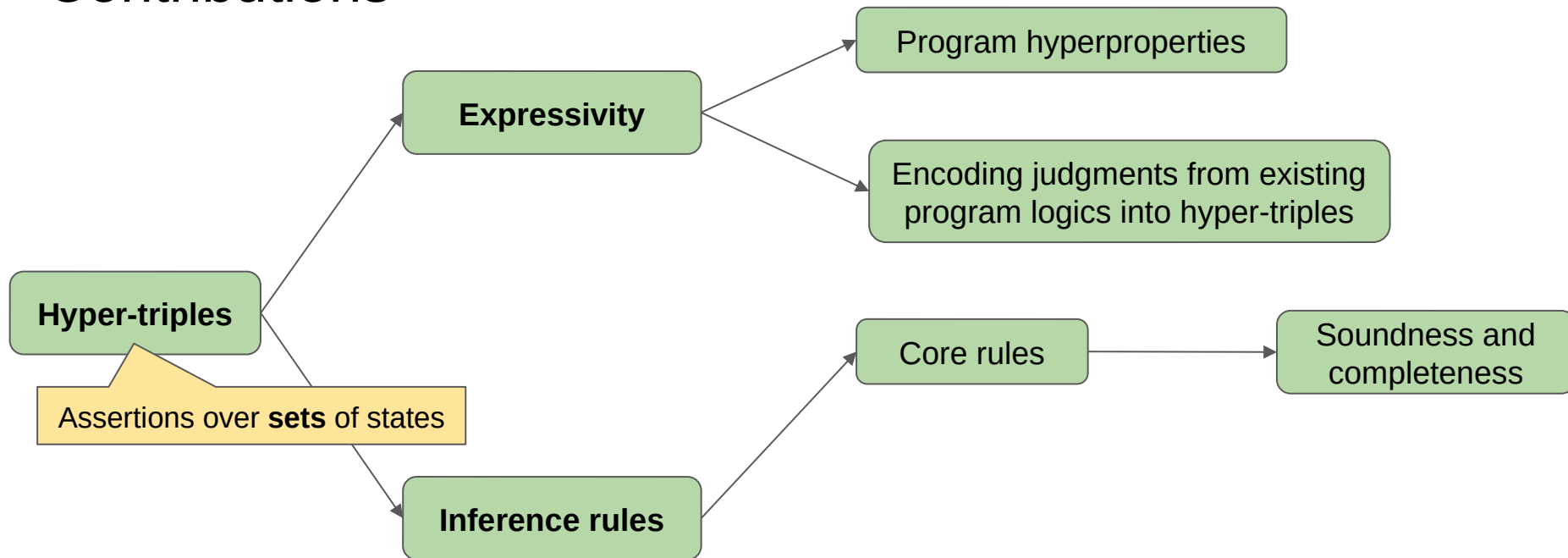
Contributions



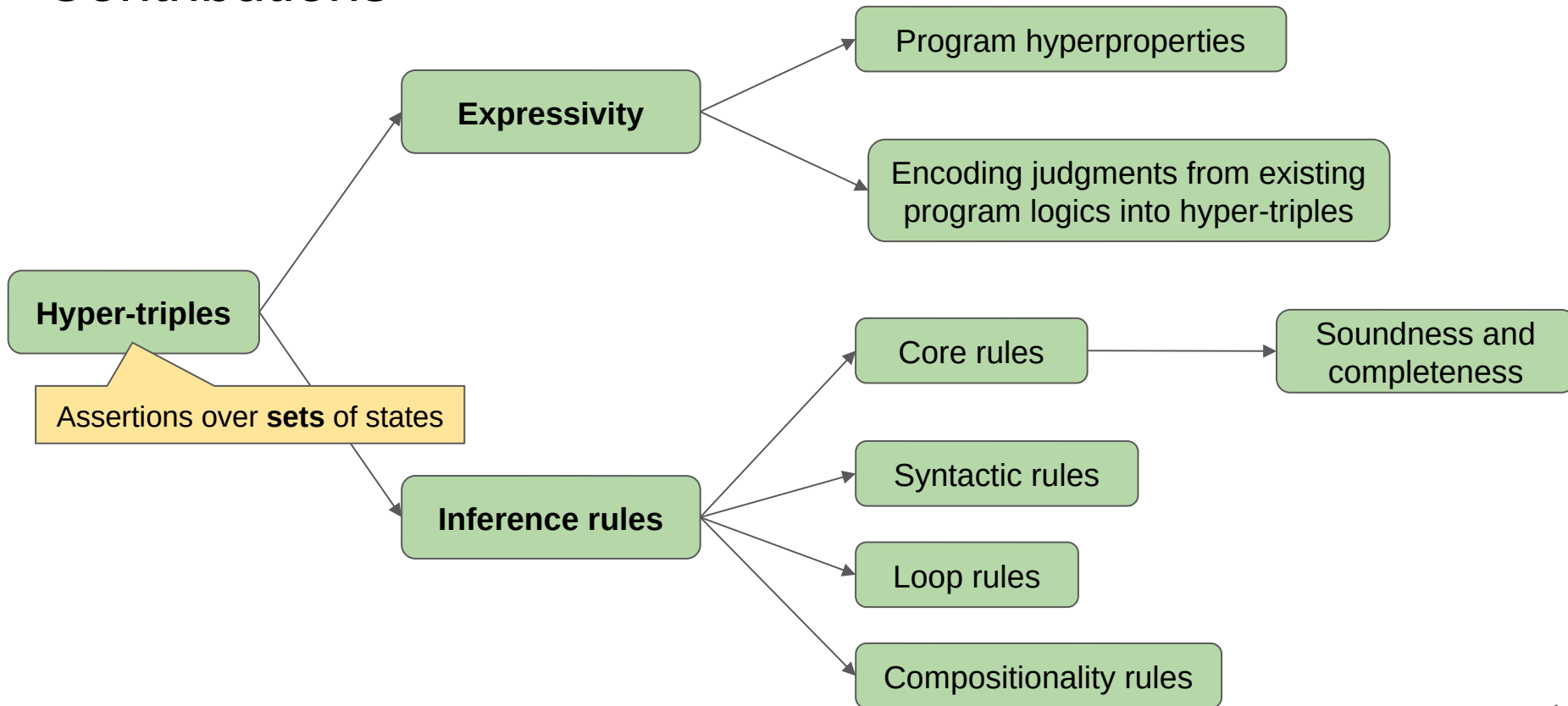
Contributions



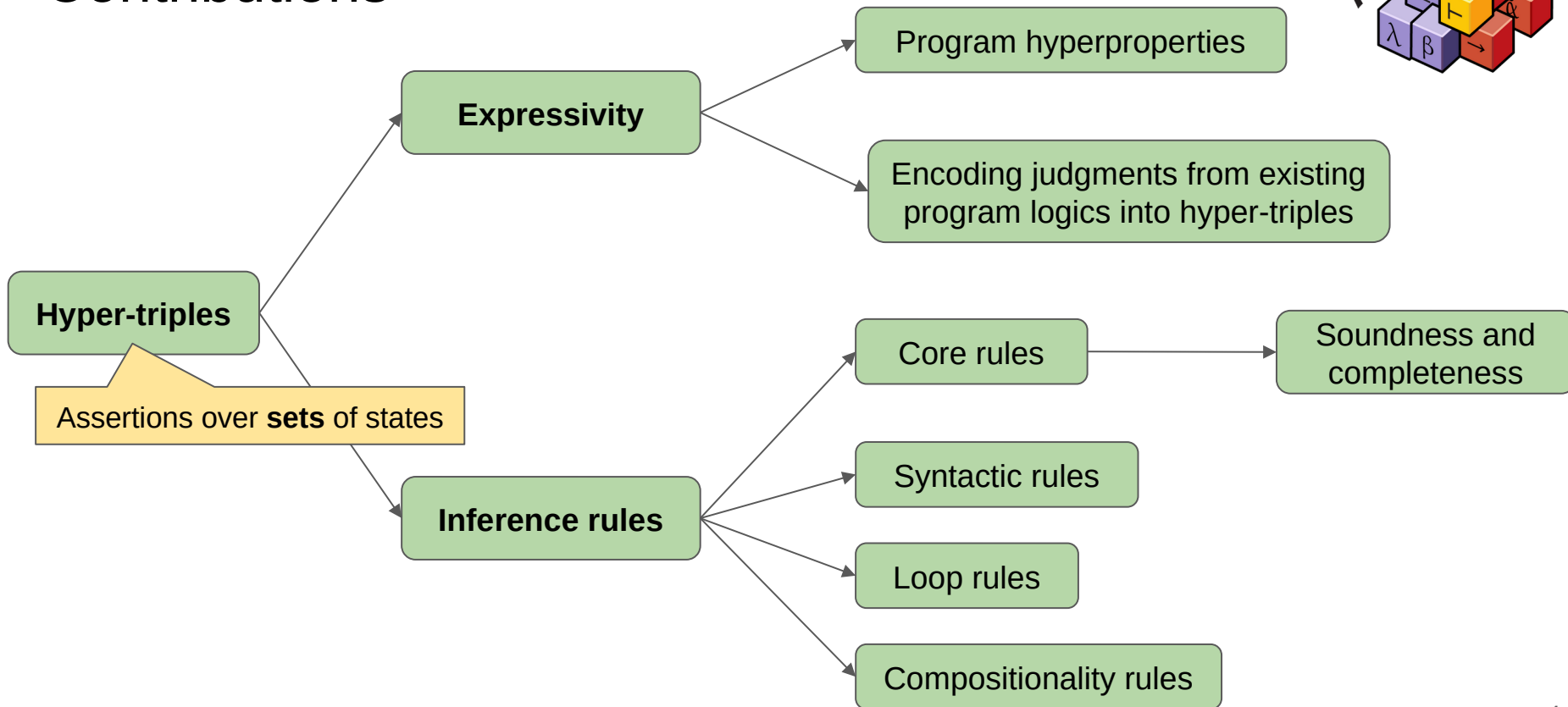
Contributions



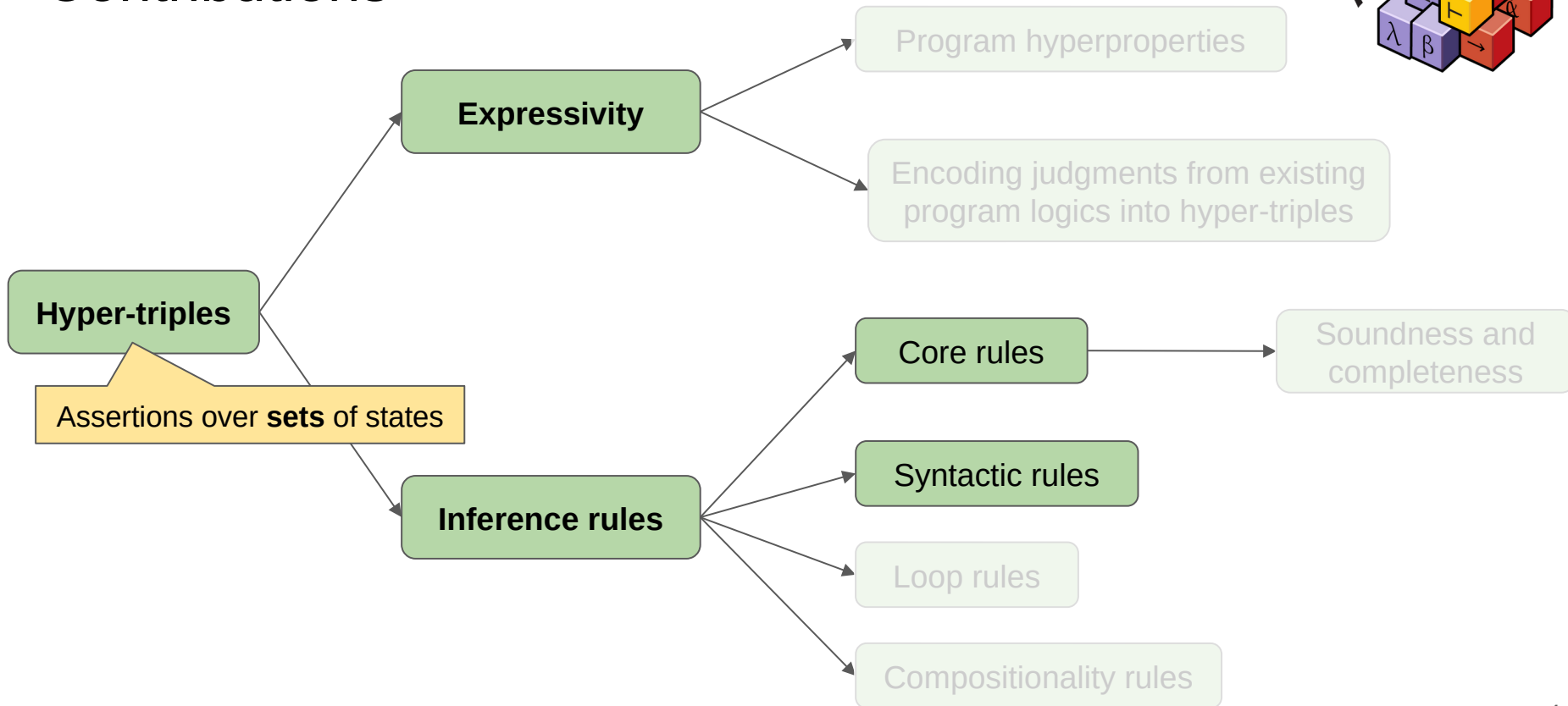
Contributions



Contributions



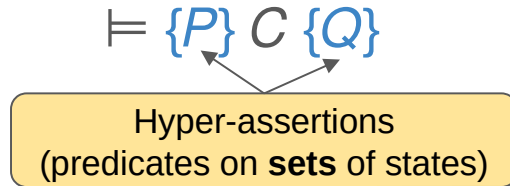
Contributions



Hyper-Triples: Tracking **Sets** of States

$$\models \{P\} C \{Q\}$$

Hyper-Triples: Tracking **Sets** of States

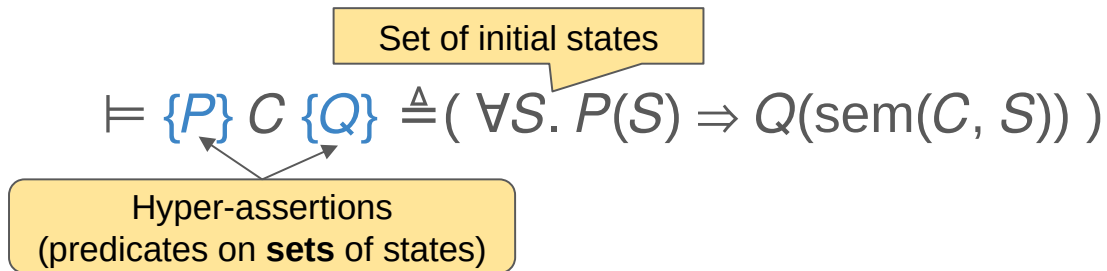


Hyper-Triples: Tracking **Sets** of States

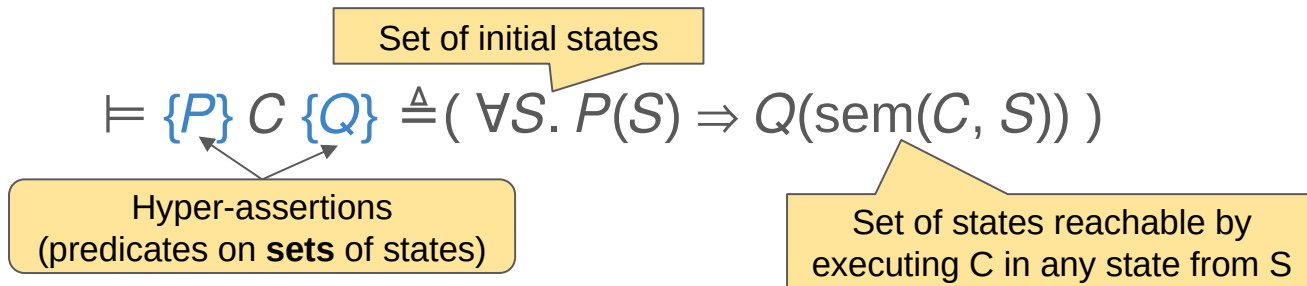
$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Hyper-assertions
(predicates on **sets** of states)

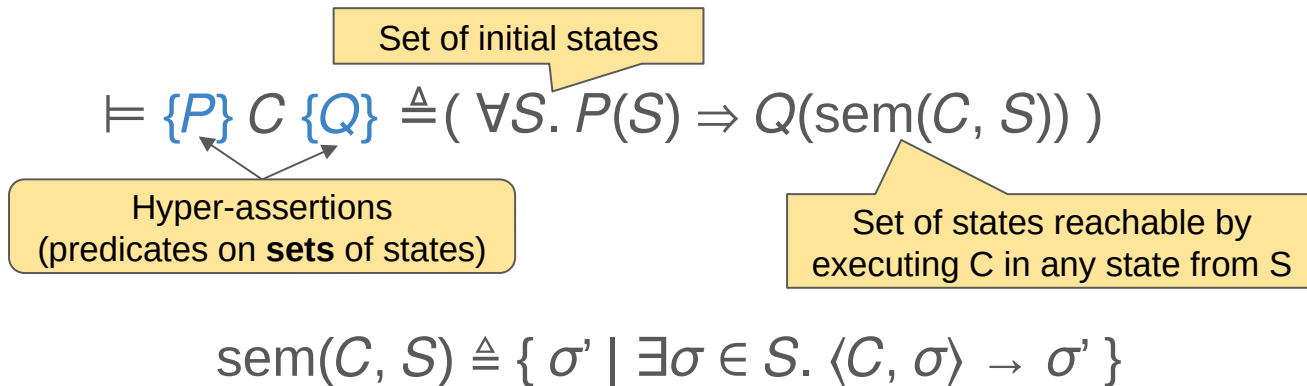
Hyper-Triples: Tracking **Sets** of States



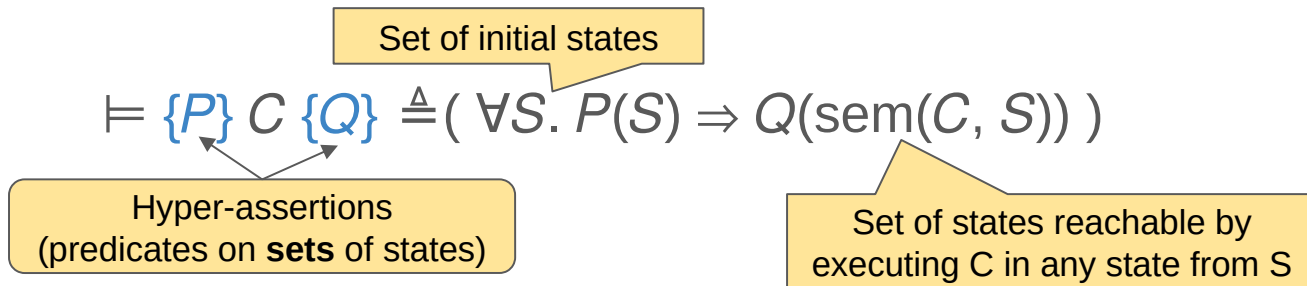
Hyper-Triples: Tracking **Sets** of States



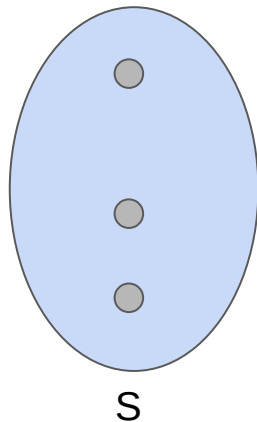
Hyper-Triples: Tracking **Sets** of States



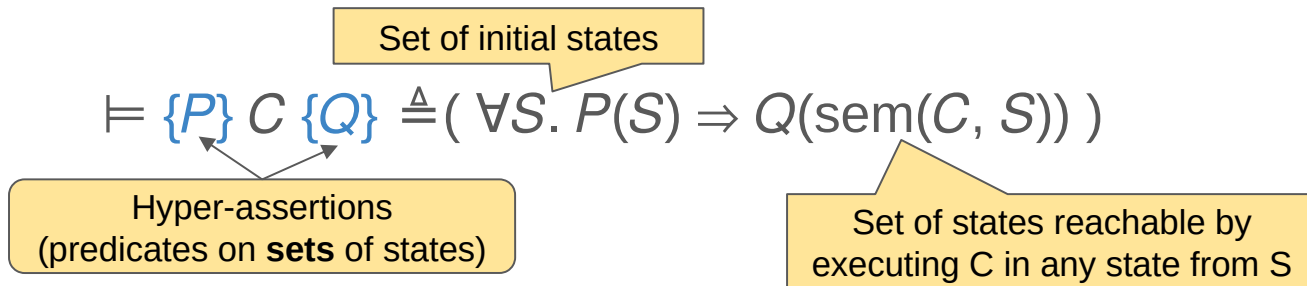
Hyper-Triples: Tracking **Sets** of States



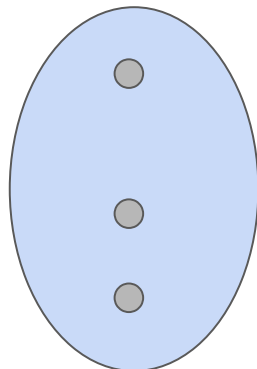
$$\text{sem}(C, S) \triangleq \{ \sigma' \mid \exists \sigma \in S. \langle C, \sigma \rangle \rightarrow \sigma' \}$$



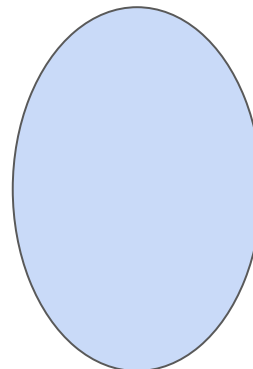
Hyper-Triples: Tracking **Sets** of States



$$\text{sem}(C, S) \triangleq \{ \sigma' \mid \exists \sigma \in S. \langle C, \sigma \rangle \rightarrow \sigma' \}$$

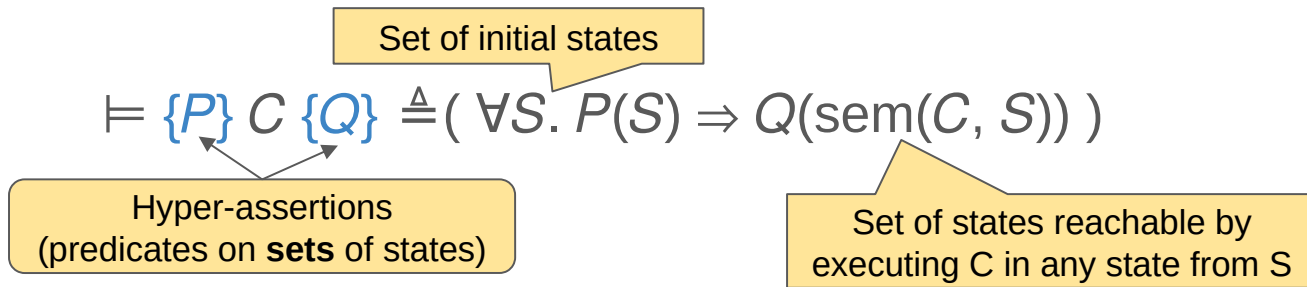


S

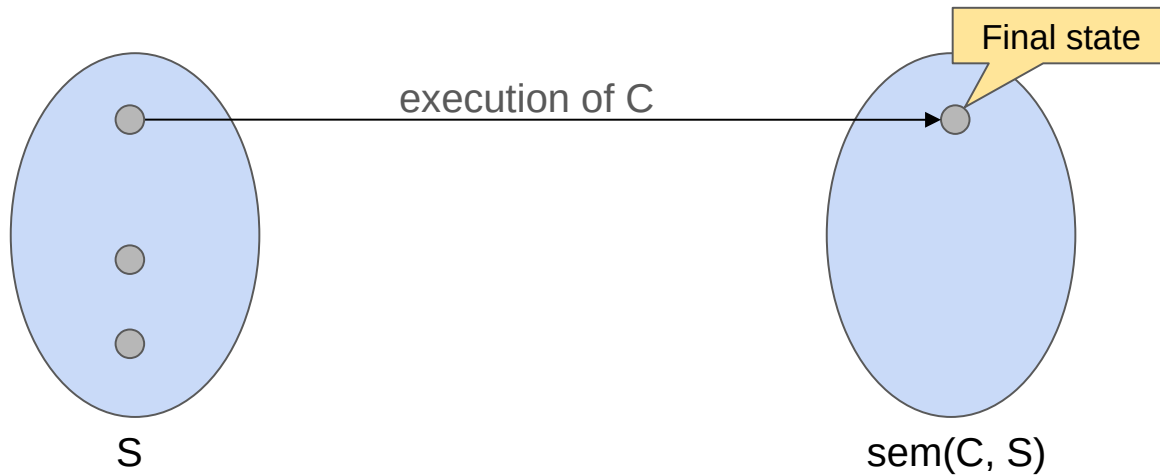


$\text{sem}(C, S)$

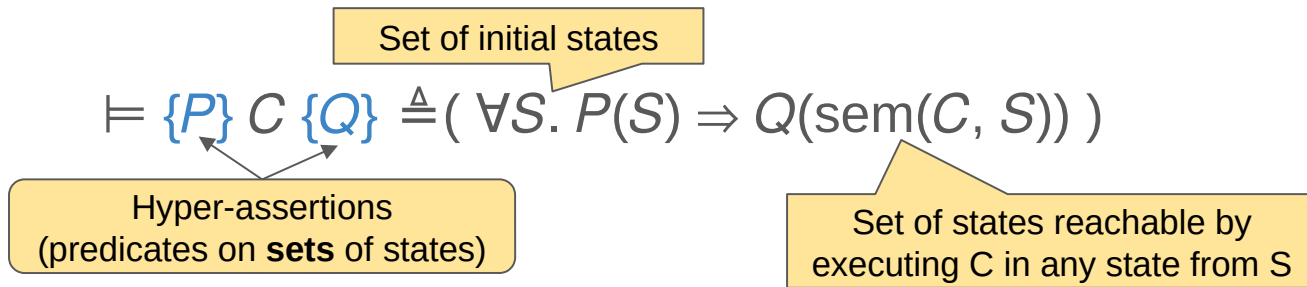
Hyper-Triples: Tracking **Sets** of States



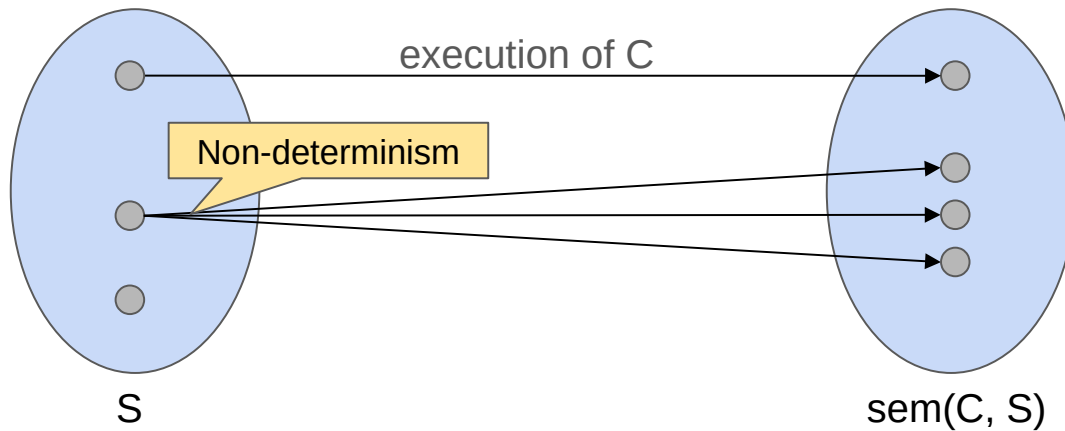
$$\text{sem}(C, S) \triangleq \{ \sigma' \mid \exists \sigma \in S. \langle C, \sigma \rangle \rightarrow \sigma' \}$$



Hyper-Triples: Tracking **Sets** of States



$$\text{sem}(C, S) \triangleq \{ \sigma' \mid \exists \sigma \in S. \langle C, \sigma \rangle \rightarrow \sigma' \}$$



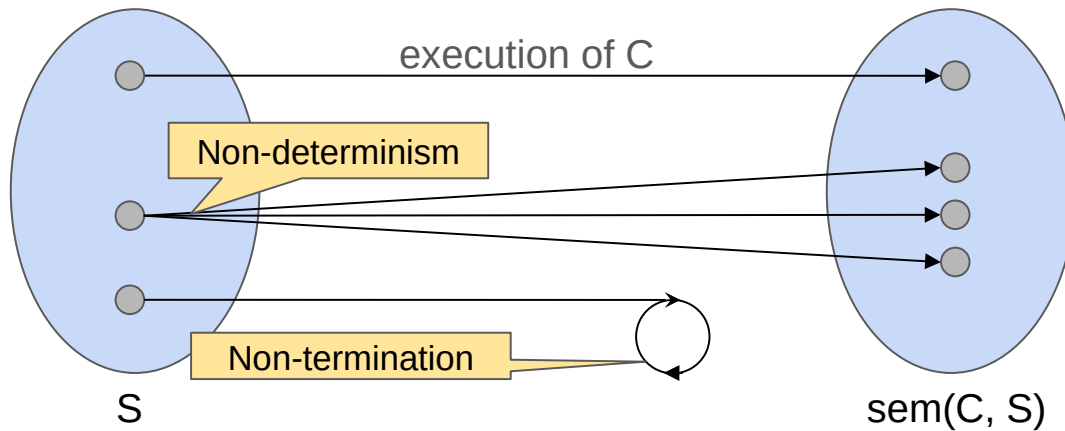
Hyper-Triples: Tracking **Sets** of States

Set of initial states

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Hyper-assertions
(predicates on **sets** of states)

$$\text{sem}(C, S) \triangleq \{ \sigma' \mid \exists \sigma \in S. \langle C, \sigma \rangle \rightarrow \sigma' \}$$



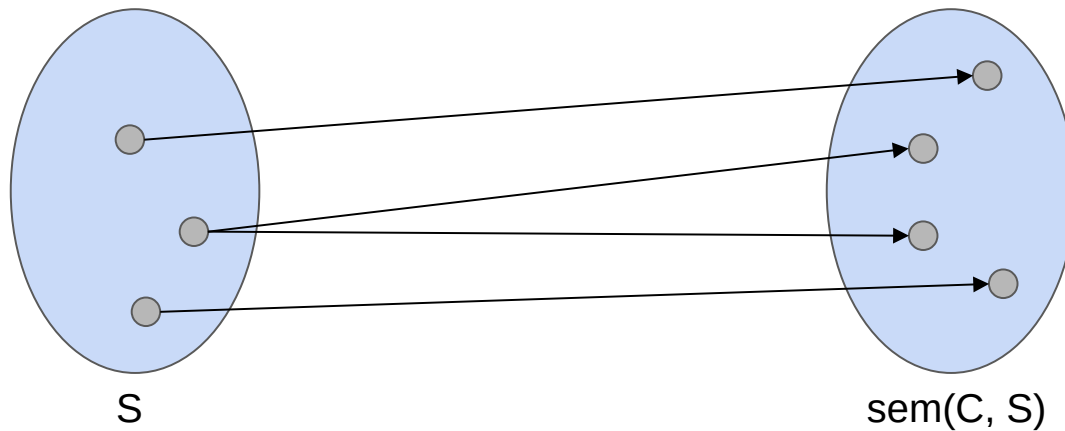
Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Non-interference (public input x, public output y)

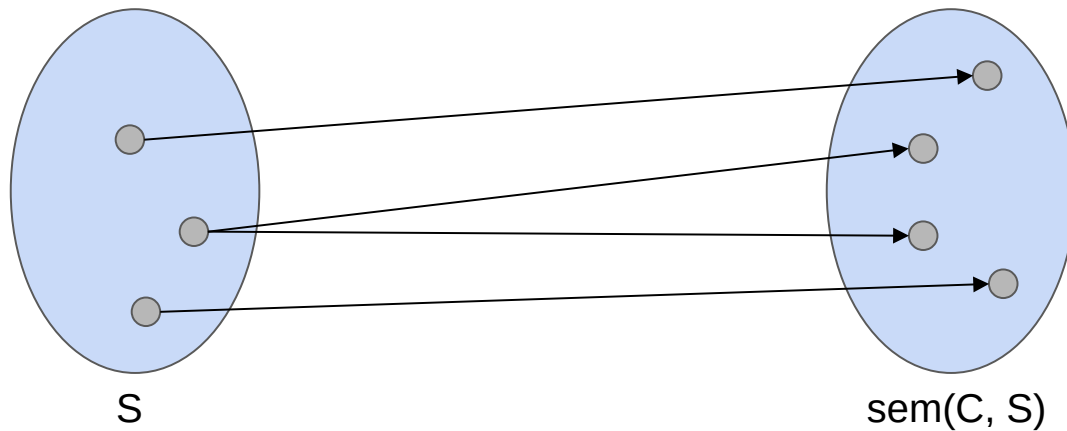


Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Non-interference (public input x, public output y)

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) = \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) = \sigma_2(y) \}$$



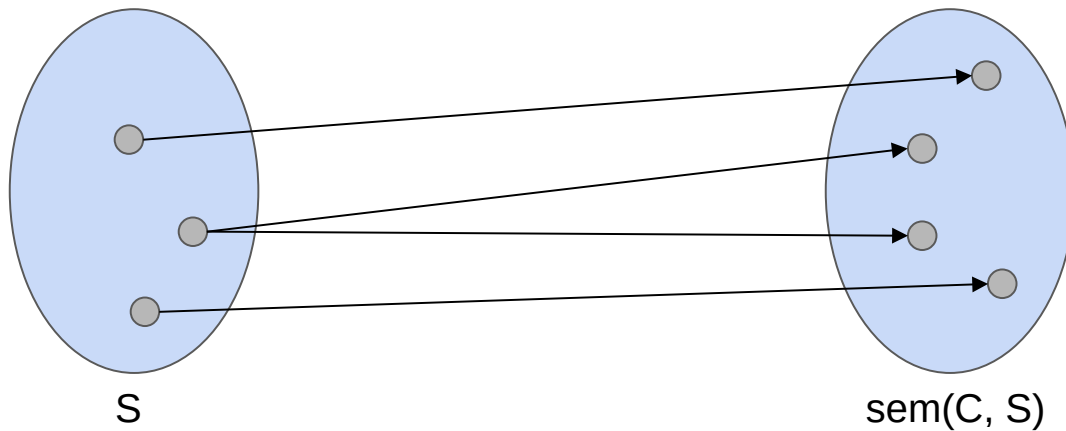
Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Non-interference (public input x, public output y)

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) = \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) = \sigma_2(y) \}$$

$$\lambda S. \forall \sigma_1 \in S. \forall \sigma_2 \in S. \sigma_1(x) = \sigma_2(x)$$



Examples

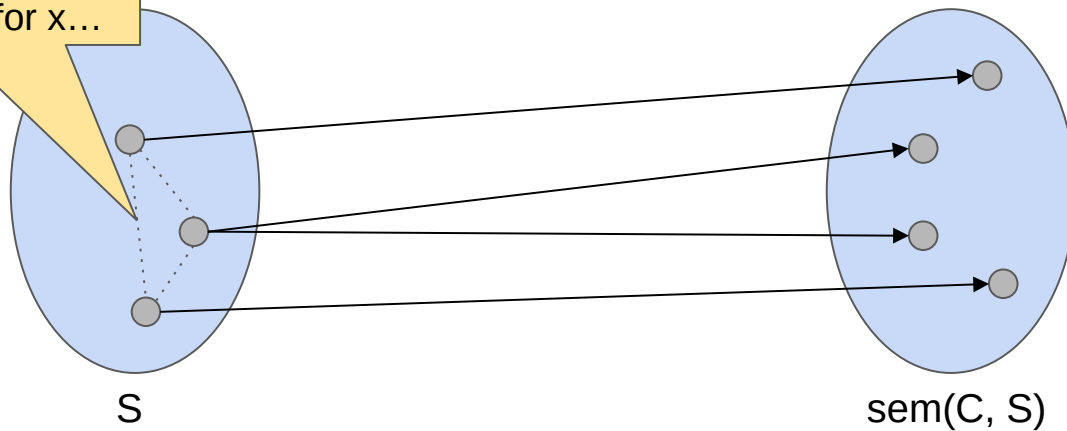
$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Non-interference (public input x, public output y)

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) = \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) = \sigma_2(y) \}$$

$$\lambda S. \forall \sigma_1 \in S. \forall \sigma_2 \in S. \sigma_1(x) = \sigma_2(x)$$

If all initial states have the same value for x...



Examples

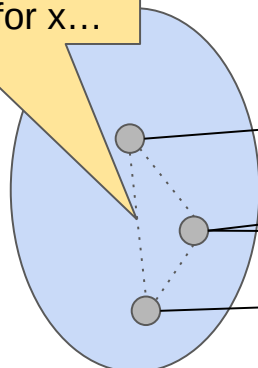
$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Non-interference (public input x, public output y)

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) = \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) = \sigma_2(y) \}$$

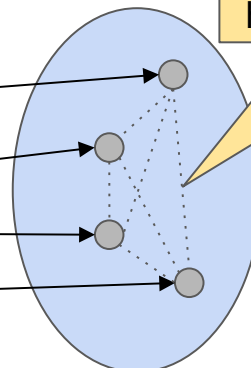
$$\lambda S. \forall \sigma_1 \in S. \forall \sigma_2 \in S. \sigma_1(x) = \sigma_2(x)$$

If all initial states have
the same value for x...



S

... then all final states
have the same value for y

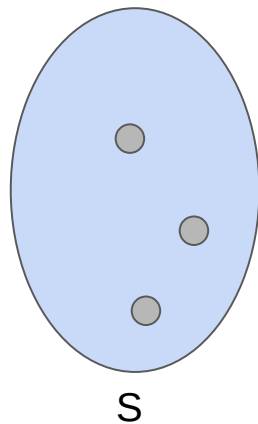


sem(C, S)

Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Monotonicity (input x, output y)

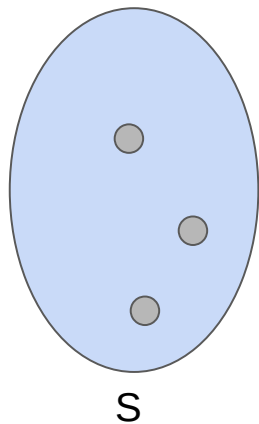


Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Monotonicity (input x, output y)

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) > \sigma_2(y) \}$$



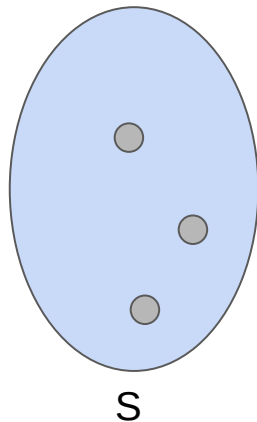
Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Monotonicity (input x, output y)

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) > \sigma_2(y) \}$$

Only satisfiable by the empty set

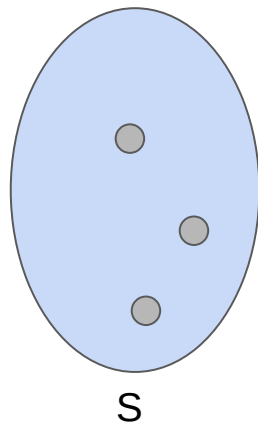


Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Monotonicity (input x, output y)

~~$$\{ \forall \sigma_1. \forall \sigma_2. \sigma_1(x) > \sigma_2(x) \} C \{ \forall \sigma_1. \forall \sigma_2. \sigma_1(y) > \sigma_2(y) \}$$~~



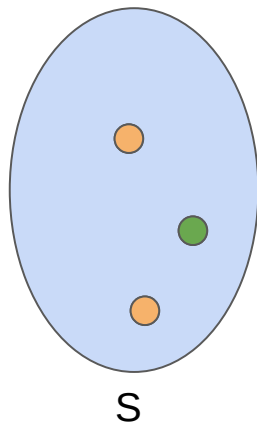
Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Monotonicity (input x, output y)

~~$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) > \sigma_2(y) \}$$~~

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(y) > \sigma_2(y) \}$$



Examples

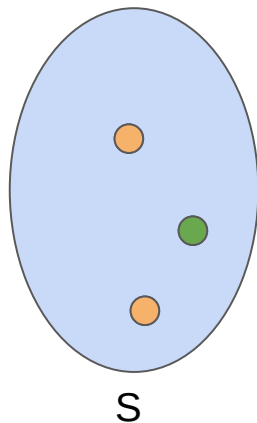
$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Monotonicity (input x, output y)

logical tag to identify executions

~~$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) > \sigma_2(y) \}$$~~

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(y) > \sigma_2(y) \}$$



Examples

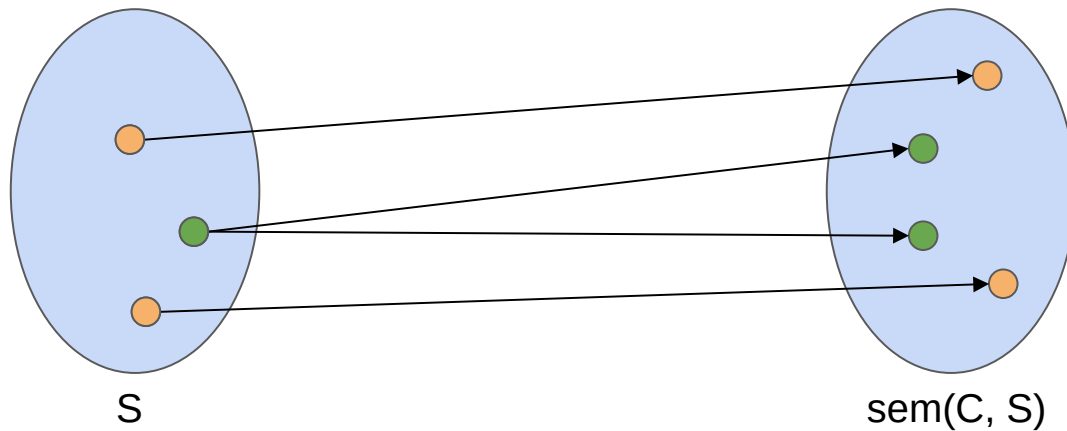
$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Monotonicity (input x, output y)

logical tag to identify executions

~~$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) > \sigma_2(y) \}$$~~

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(y) > \sigma_2(y) \}$$



Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

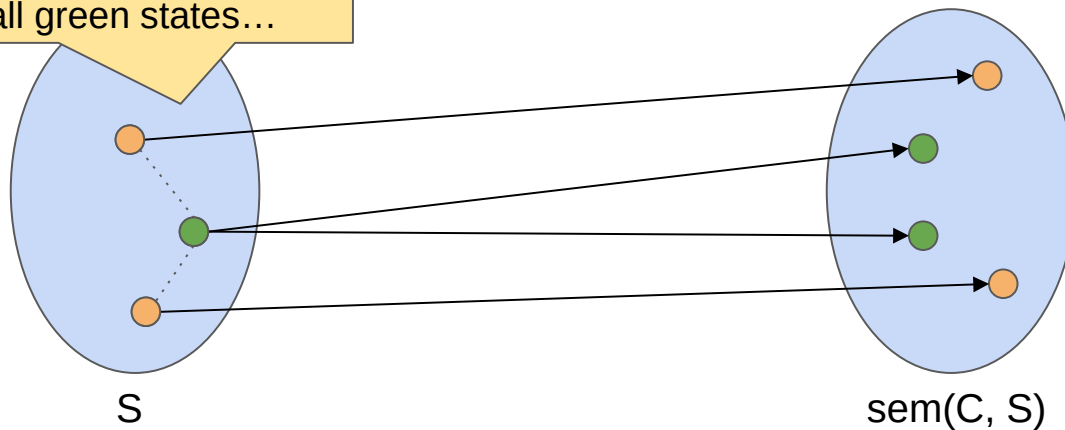
Monotonicity (input x, output y)

logical tag to identify executions

~~$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) > \sigma_2(y) \}$$~~

$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(y) > \sigma_2(y) \}$$

If all initial orange states have a larger value for x than all green states...



Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

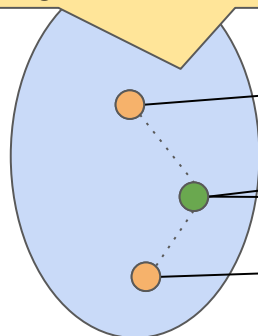
Monotonicity (input x, output y)

logical tag to identify executions

~~$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(y) > \sigma_2(y) \}$$~~

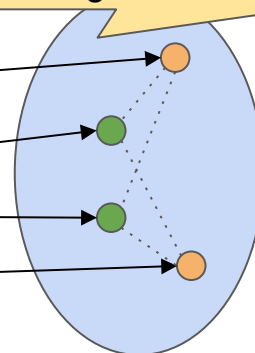
$$\{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(x) > \sigma_2(x) \} C \{ \forall \langle \sigma_1 \rangle. \forall \langle \sigma_2 \rangle. \sigma_1(t)=1 \wedge \sigma_2(t)=2 \Rightarrow \sigma_1(y) > \sigma_2(y) \}$$

If all initial orange states have a larger value for x than all green states...



S

... then all final orange states have a larger value for y than all green states

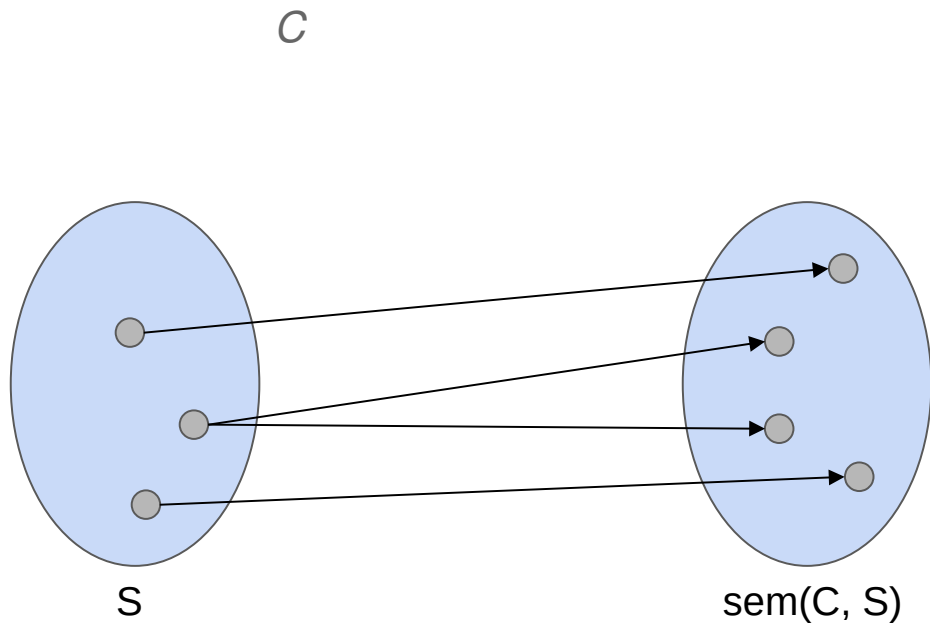


sem(C, S)

Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Existence of a state with minimal values (for x and y)

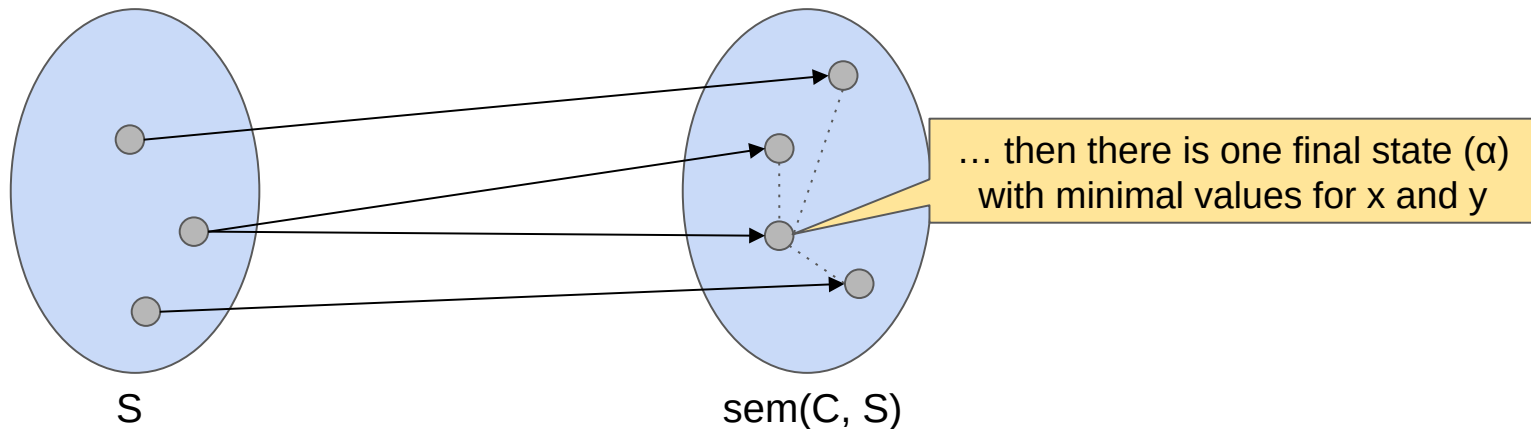


Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Existence of a state with minimal values (for x and y)

$$C \{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$$



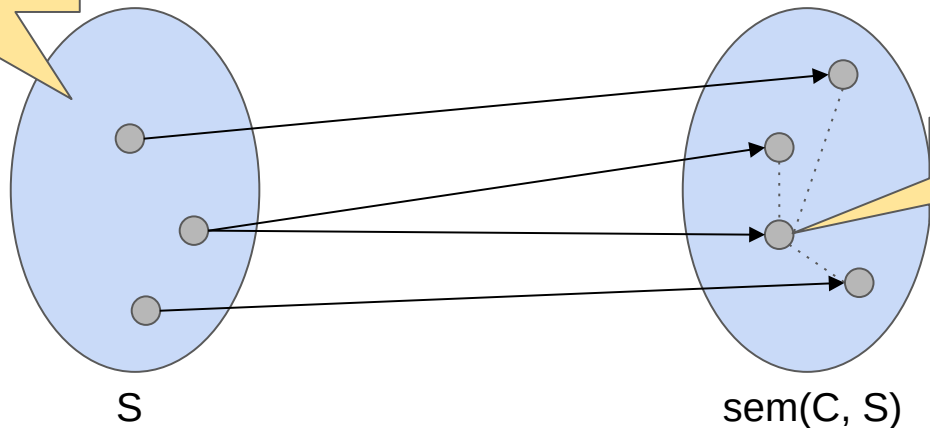
Examples

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

Existence of a state with minimal values (for x and y)

$$\{ \exists \langle \alpha \rangle \} C \{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$$

If there is at least
one initial state...



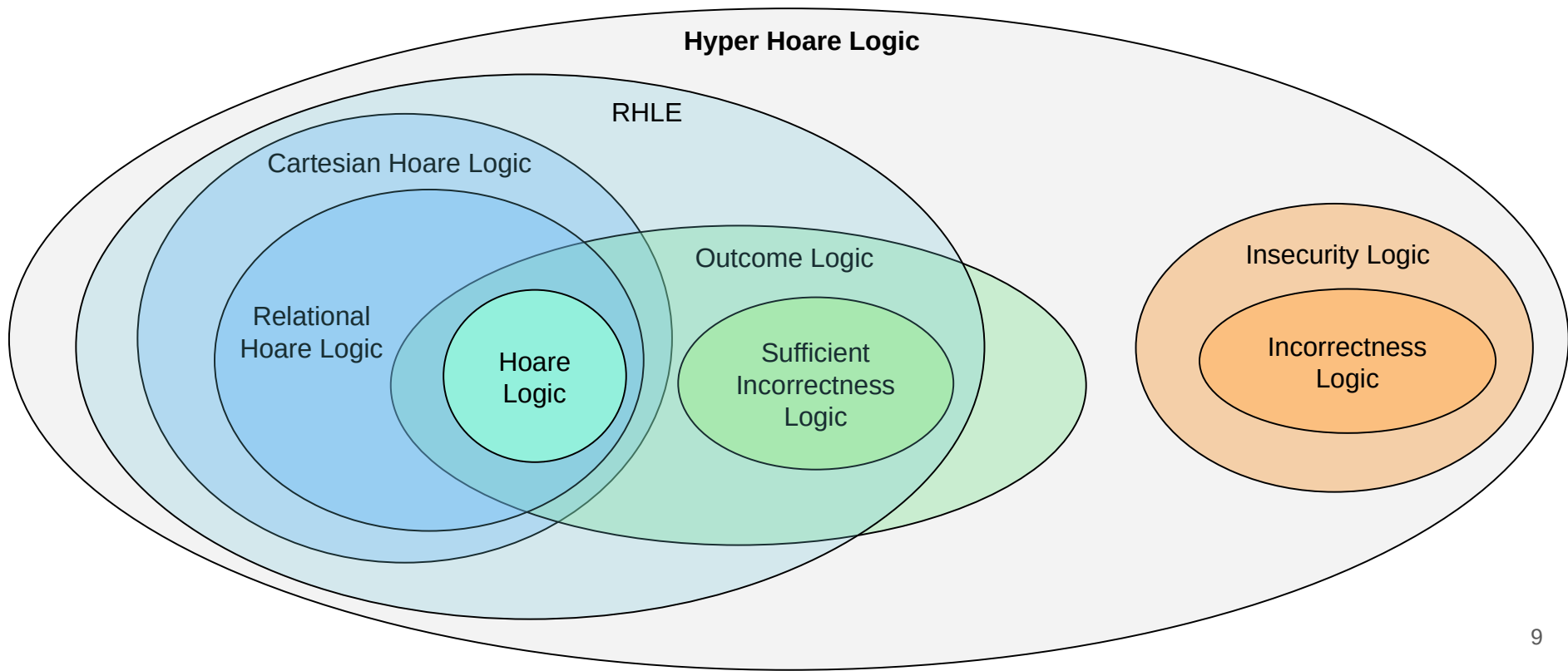
... then there is one final state (α)
with minimal values for x and y

Expressivity: Judgments from Existing Logics

expressivity of judgments restricted to a **single non-deterministic IMP** program (no heap or probabilities)

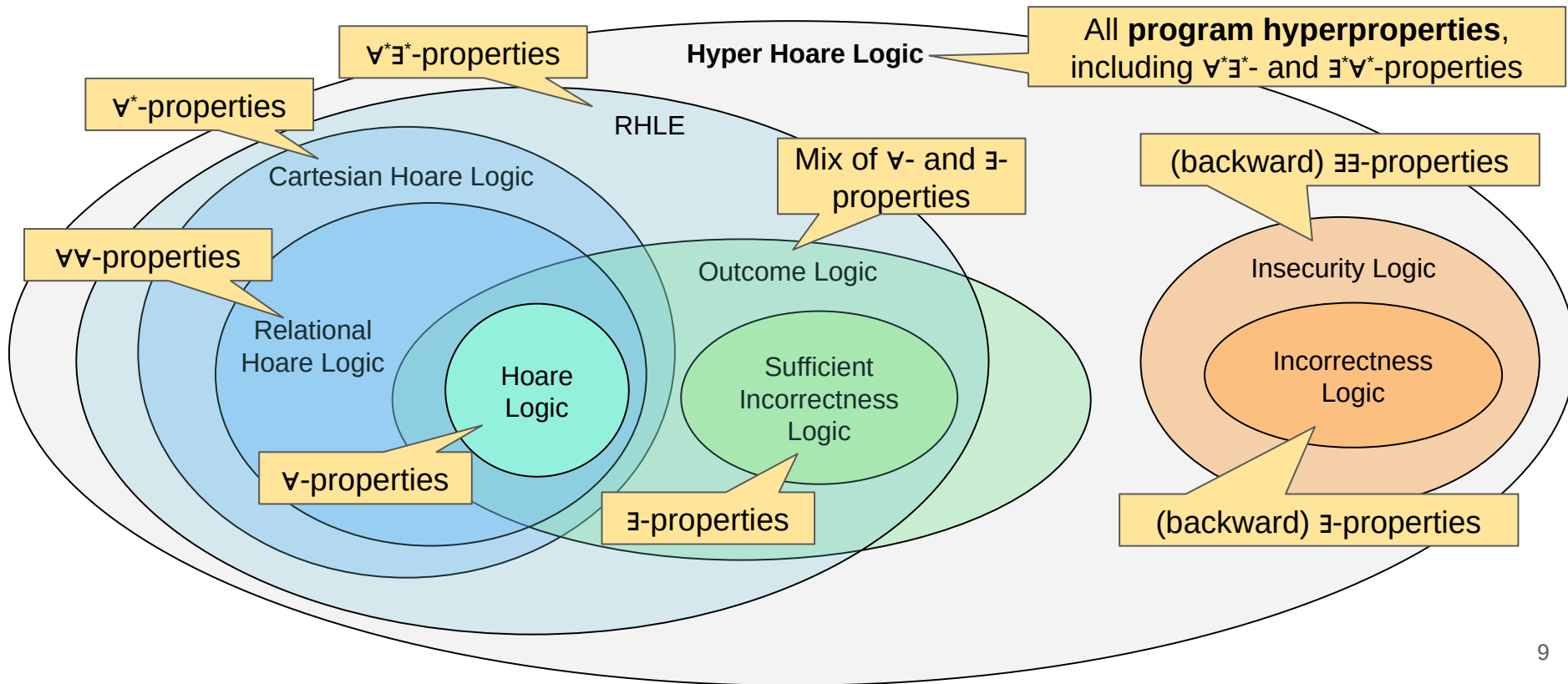
Expressivity: Judgments from Existing Logics

expressivity of judgments restricted to a **single non-deterministic IMP** program (no heap or probabilities)



Expressivity: Judgments from Existing Logics

expressivity of judgments restricted to a **single non-deterministic IMP** program (no heap or probabilities)



Core Rules (Sound and Complete)

Core Rules (Sound and Complete)

$C ::= \mathbf{skip} \mid x := e \mid x := \text{nonDet}() \mid \mathbf{assume} \ b \mid C; C \mid C + C \mid C^*$

Core Rules (Sound and Complete)

$C ::= \mathbf{skip} \mid x := e \mid x := \mathbf{nonDet}() \mid \mathbf{assume} \ b \mid C; C \mid C + C \mid C^*$



Non-deterministic assignment

Core Rules (Sound and Complete)

$C ::= \mathbf{skip} \mid x := e \mid x := \mathbf{nonDet}() \mid \mathbf{assume} \ b \mid C; C \mid C + C \mid C^*$



Non-deterministic assignment



Stops the execution if b does not hold

Core Rules (Sound and Complete)

$C ::= \mathbf{skip} \mid x := e \mid x := \text{nonDet}() \mid \mathbf{assume} \ b \mid C; C \mid C + C \mid C^*$



Non-deterministic choice



Non-deterministic iteration

Core Rules (Sound and Complete)

$C ::= \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } b \mid C; C \mid C + C \mid C^*$



Non-deterministic choice



Non-deterministic iteration

$\text{if } (b) \{ C \} \text{ else } \{ C' \} \triangleq (\text{assume } b; C) + (\text{assume } \neg b; C')$

$\text{while } (b) \{ C \} \triangleq (\text{assume } b; C)^* ; \text{assume } \neg b$

Core Rules (Sound and Complete)

$C ::= \mathbf{skip} \mid x := e \mid x := \text{nonDet}() \mid \mathbf{assume} \ b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

Core Rules (Sound and Complete)

$C ::= \mathbf{skip} \mid x := e \mid x := \text{nonDet}() \mid \mathbf{assume} \ b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

$$\frac{\vdash \{P\} C_1 \{R\} \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}} \text{ (Seq)}$$

Core Rules (Sound and Complete)

$C ::= \mathbf{skip} \mid x := e \mid x := \text{nonDet}() \mid \mathbf{assume} \ b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

Needed for completeness

$$\frac{\vdash \{P\} C_1 \{R\} \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}} \text{ (Seq)}$$

Core Rules (Sound and Complete)

$C ::= \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

Needed for completeness

$$\frac{\vdash \{P\} C_1 \{R\} \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}} \text{ (Seq)}$$

2 branching rules

choice, iteration

Core Rules (Sound and Complete)

$C ::= \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

Needed for completeness

$$\frac{\vdash \{P\} C_1 \{R\} \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}} \text{ (Seq)}$$

2 branching rules

choice, iteration

$$\frac{\{P\} C_1 \{Q_1\} \quad \vdash \{P\} C_2 \{Q_2\}}{\vdash \{P\} C_1 + C_2 \{Q_1 \otimes Q_2\}} \text{ (Choice)}$$

Core Rules (Sound and Complete)

$C ::= \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

Needed for completeness

$$\frac{\vdash \{P\} C_1 \{R\} \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}} \text{ (Seq)}$$

2 branching rules

choice, iteration

$$\frac{\{P\} C_1 \{Q_1\} \quad \vdash \{P\} C_2 \{Q_2\}}{\vdash \{P\} C_1 + C_2 \{Q_1 \otimes Q_2\}} \text{ (Choice)}$$

$\exists S_1, S_2. S = S_1 \cup S_2 \wedge Q_1(S_1) \wedge Q_2(S_2)$

Core Rules (Sound and Complete)

$C ::= \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

Needed for completeness

$$\frac{\vdash \{P\} C_1 \{R\} \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}} \text{ (Seq)}$$

2 branching rules

choice, iteration

$$\frac{\{P\} C_1 \{Q_1\} \quad \vdash \{P\} C_2 \{Q_2\}}{\vdash \{P\} C_1 + C_2 \{Q_1 \otimes Q_2\}} \text{ (Choice)}$$

$\exists S_1, S_2. S = S_1 \cup S_2 \wedge Q_1(S_1) \wedge Q_2(S_2)$

3 atomic rules

assignments, assume

Core Rules (Sound and Complete)

$C ::= \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

Needed for completeness

$$\frac{\vdash \{P\} C_1 \{R\} \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}} \text{ (Seq)}$$

2 branching rules

choice, iteration

$$\frac{\{P\} C_1 \{Q_1\} \quad \vdash \{P\} C_2 \{Q_2\}}{\vdash \{P\} C_1 + C_2 \{Q_1 \otimes Q_2\}} \text{ (Choice)}$$

$\exists S_1, S_2. S = S_1 \cup S_2 \wedge Q_1(S_1) \wedge Q_2(S_2)$

3 atomic rules

assignments, assume

$$\frac{}{\vdash \{\lambda S. P(\{\sigma' \mid \exists \sigma \in S. \sigma' = \sigma[x \mapsto e(\sigma)]\})\} x := e \{P\}} \text{ (Assign)}$$

Core Rules (Sound and Complete)

$C ::= \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

Needed for completeness

$$\frac{\vdash \{P\} C_1 \{R\} \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}} \text{ (Seq)}$$

2 branching rules

choice, iteration

$$\frac{\{P\} C_1 \{Q_1\} \quad \vdash \{P\} C_2 \{Q_2\}}{\vdash \{P\} C_1 + C_2 \{Q_1 \otimes Q_2\}} \text{ (Choice)}$$

$\exists S_1, S_2. S = S_1 \cup S_2 \wedge Q_1(S_1) \wedge Q_2(S_2)$

3 atomic rules

assignments, assume

$$\frac{}{\vdash \underbrace{\{\lambda S. P(\{\sigma' \mid \exists \sigma \in S. \sigma' = \sigma[x \mapsto e(\sigma)]\})\}}_{\text{sem}(x := e, S)} x := e \{P\}} \text{ (Assign)}$$

Core Rules (Sound and Complete)

$C ::= \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } b \mid C; C \mid C + C \mid C^*$

4 standard rules

seq, skip, consequence, exist

Needed for completeness

$$\frac{\vdash \{P\} C_1 \{R\} \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}} \text{ (Seq)}$$

2 branching rules

choice, iteration

$$\frac{\{P\} C_1 \{Q_1\} \quad \vdash \{P\} C_2 \{Q_2\}}{\vdash \{P\} C_1 + C_2 \{Q_1 \otimes Q_2\}} \text{ (Choice)}$$

Can be seen as a **syntactic substitution**
(for syntactic hyper-assertions)

$$\exists S_1, S_2. S = S_1 \cup S_2 \wedge Q_1(S_1) \wedge Q_2(S_2)$$

3 atomic rules

assignments, assume

$$\frac{}{\vdash \{ \lambda S. P(\{ \sigma' \mid \exists \sigma \in S. \sigma' = \sigma[x \mapsto e(\sigma)] \}) \} x := e \{P\}} \text{ (Assign)}$$

$\text{sem}(x := e, S)$

Syntactic (Backward) Rules

$$A ::= b \mid e \geq e \mid A \vee A \mid A \wedge A \mid \forall y. A \mid \exists y. A \mid \forall \langle \varphi \rangle. A \mid \exists \langle \varphi \rangle. A$$

Syntactic (Backward) Rules

$A ::= b \mid e \geq e \mid A \vee A \mid A \wedge A \mid \forall y. A \mid \exists y. A \mid \forall \langle \varphi \rangle. A \mid \exists \langle \varphi \rangle. A$



Restricted interaction
with the set of states

Syntactic (Backward) Rules

$A ::= b \mid e \geq e \mid A \vee A \mid A \wedge A \mid \forall y. A \mid \exists y. A \mid \forall \langle \varphi \rangle. A \mid \exists \langle \varphi \rangle. A$

Restricted interaction
with the set of states

$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

assume $r \geq 0$

$y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

$$A ::= b \mid e \geq e \mid A \vee A \mid A \wedge A \mid \forall y. A \mid \exists y. A \mid \forall \langle \varphi \rangle. A \mid \exists \langle \varphi \rangle. A$$

If there exists an initial state α with $r = 0$ and minimal value for x

$$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$$

assume $r \geq 0$

$y := x + r$

$$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$$

Syntactic (Backward) Rules

$A ::= b \mid e \geq e \mid A \vee A \mid A \wedge A \mid \forall y. A \mid \exists y. A \mid \forall \langle \varphi \rangle. A \mid \exists \langle \varphi \rangle. A$

If there exists an initial state α with $r = 0$ and minimal value for x

$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

assume $r \geq 0$

$y := x + r$

Then there exists a final state α with minimal values for x and y

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

assume $r \geq 0$

 $y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

$$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$$

assume $r \geq 0$

➔ $y := x + r$

$$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \boxed{\alpha(y)} \leq \boxed{\sigma(y)} \}$$

Syntactic (Backward) Rules

$$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$$

assume $r \geq 0$

$$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \boxed{\alpha(x) + \alpha(r)} \leq \boxed{\sigma(x) + \sigma(r)} \}$$

→ $\boxed{y} := \boxed{x + r}$

$$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \boxed{\alpha(y)} \leq \boxed{\sigma(y)} \}$$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

$\triangleright \sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

Syntactic substitution

$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

assume $r \geq 0$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \boxed{\alpha(x) + \alpha(r)} \leq \boxed{\sigma(x) + \sigma(r)} \}$

$\Rightarrow \boxed{y} := \boxed{x + r}$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \boxed{\alpha(y)} \leq \boxed{\sigma(y)} \}$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

➤ $\sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

➡ **assume** $r \geq 0$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r) \}$

$y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

➤ $\sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

➡ **assume** $r \geq 0$

$\{ \boxed{\exists \langle \alpha \rangle} \boxed{\forall \langle \sigma \rangle}. \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r) \}$

$y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

➤ $\sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

$\{ \boxed{\exists \langle \alpha \rangle. \alpha(r) \geq 0} \wedge$

➡ **assume** $r \geq 0$

$\{ \boxed{\exists \langle \alpha \rangle} \boxed{\forall \langle \sigma \rangle}. \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r) \}$

$y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

➤ $\sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

Proof obligation

$\{ \exists \langle \alpha \rangle. \alpha(r) \geq 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

$\{ \exists \langle \alpha \rangle. \alpha(r) \geq 0 \wedge$

➔ **assume** $r \geq 0$

$\{ \exists \langle \alpha \rangle \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r) \}$

$y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

➤ $\sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

Proof obligation

$\{ \exists \langle \alpha \rangle. \alpha(r) \geq 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

$\{ \boxed{\exists \langle \alpha \rangle. \alpha(r) \geq 0} \wedge \boxed{(\forall \langle \sigma \rangle. \sigma(r) \geq 0 \Rightarrow$

➡ **assume** $r \geq 0$

$\{ \boxed{\exists \langle \alpha \rangle} \boxed{\forall \langle \sigma \rangle}. \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r) \}$

$y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

➤ $\sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

Proof obligation

Assumption

$\{ \exists \langle \alpha \rangle. \alpha(r) \geq 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

$\{ \boxed{\exists \langle \alpha \rangle. \alpha(r) \geq 0} \wedge \boxed{(\forall \langle \sigma \rangle. \sigma(r) \geq 0 \Rightarrow}$

➡ **assume** $r \geq 0$

$\{ \boxed{\exists \langle \alpha \rangle} \boxed{\forall \langle \sigma \rangle}. \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r) \}$

$y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

➤ $\sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

Proof obligation

Assumption

$\{ \exists \langle \alpha \rangle. \alpha(r) \geq 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

$\{ \boxed{\exists \langle \alpha \rangle. \alpha(r) \geq 0} \wedge (\boxed{\forall \langle \sigma \rangle. \sigma(r) \geq 0 \Rightarrow} \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r)) \}$

➡ **assume** $r \geq 0$

$\{ \boxed{\exists \langle \alpha \rangle} \boxed{\forall \langle \sigma \rangle.} \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r) \}$

$y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

➤ $\sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

Precondition for **assume** b

➤ $\forall \langle \sigma \rangle. P \rightsquigarrow \forall \langle \sigma \rangle. b(\sigma) \Rightarrow P$
 ➤ $\exists \langle \sigma \rangle. P \rightsquigarrow \exists \langle \sigma \rangle. b(\sigma) \wedge P$

Proof obligation

Assumption

$\{ \exists \langle \alpha \rangle. \alpha(r) \geq 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$

$\{ \boxed{\exists \langle \alpha \rangle. \alpha(r) \geq 0} \wedge \boxed{(\forall \langle \sigma \rangle. \sigma(r) \geq 0 \Rightarrow \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r))} \}$

➡ **assume** $r \geq 0$

$\{ \boxed{\exists \langle \alpha \rangle} \boxed{\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r)} \}$

$y := x + r$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Syntactic (Backward) Rules

Precondition for **assignment** $y := e$

➤ $\sigma(y) \rightsquigarrow e(\sigma)$ (for all quantified σ)

Precondition for **assume** b

➤ $\forall \langle \sigma \rangle. P \rightsquigarrow \forall \langle \sigma \rangle. b(\sigma) \Rightarrow P$
➤ $\exists \langle \sigma \rangle. P \rightsquigarrow \exists \langle \sigma \rangle. b(\sigma) \wedge P$

Consequence rule

$\{ \exists \langle \alpha \rangle. \alpha(r) = 0 \wedge (\forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x)) \}$



$\{ \exists \langle \alpha \rangle. \alpha(r) \geq 0 \wedge (\forall \langle \sigma \rangle. \sigma(r) \geq 0 \Rightarrow \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r)) \}$

assume $r \geq 0$

$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(x) + \alpha(r) \leq \sigma(x) + \sigma(r) \}$

$y := x + r$

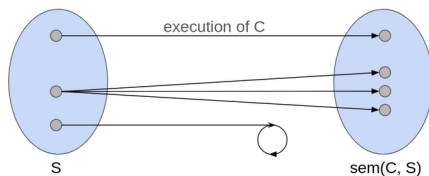
$\{ \exists \langle \alpha \rangle. \forall \langle \sigma \rangle. \alpha(x) \leq \sigma(x) \wedge \alpha(y) \leq \sigma(y) \}$

Hyper-Triples: Tracking **Sets** of States

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

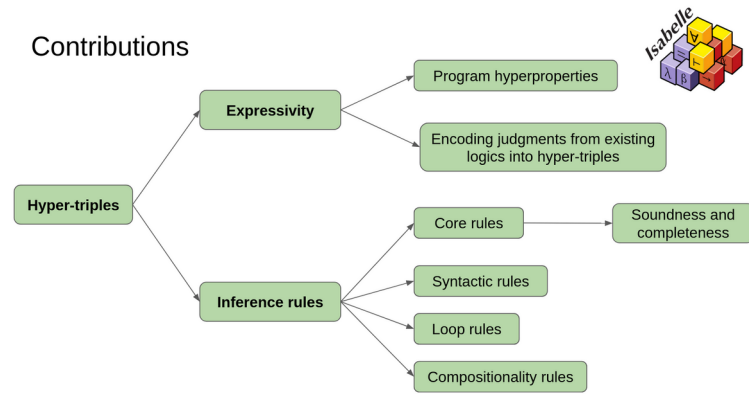
Hyper-assertions
(predicates on **sets** of states)

$$\text{sem}(C, S) \triangleq \{ \sigma' \mid \exists \sigma \in S. \langle C, \sigma \rangle \rightarrow \sigma' \}$$



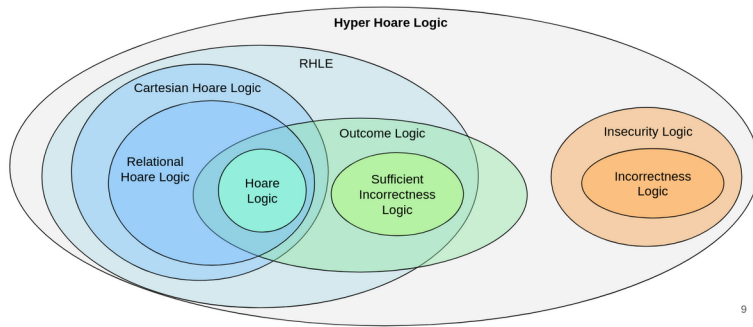
5

Contributions



Expressivity: Judgments from Existing Logics

expressivity of judgments for a **single non-deterministic IMP** program (no heap, no probabilities)



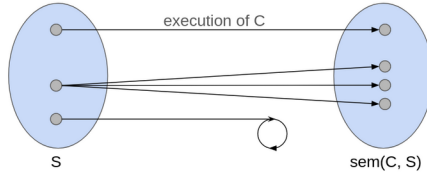
9

Hyper-Triples: Tracking **Sets** of States

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

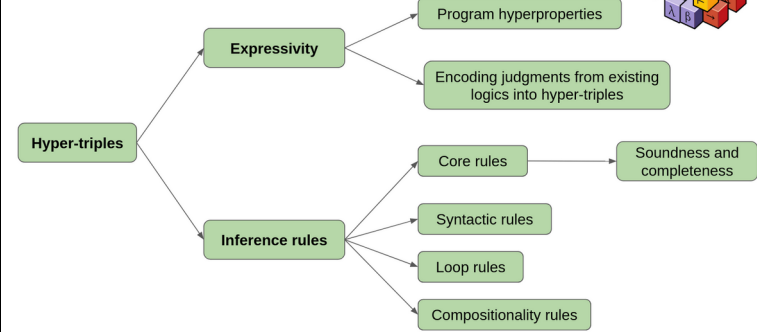
Hyper-assertions
(predicates on **sets** of states)

$$\text{sem}(C, S) \triangleq \{ \sigma' \mid \exists \sigma \in S. \langle C, \sigma \rangle \rightarrow \sigma' \}$$



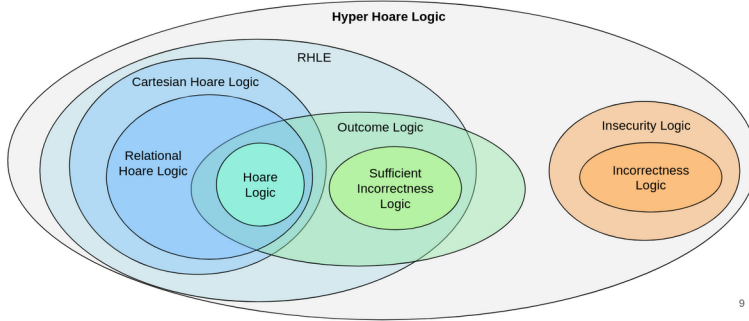
5

Contributions



Expressivity: Judgments from Existing Logics

expressivity of judgments for a **single non-deterministic IMP** program (no heap, no probabilities)



9

Future work

- Automation
- Extension to relational properties (more than one program)
- Extension to heap-manipulating programs

Thank you for your attention!

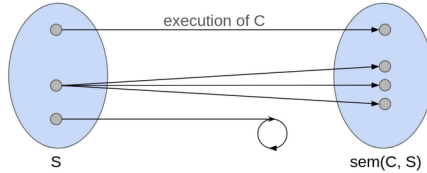


Hyper-Triples: Tracking **Sets** of States

$$\models \{P\} C \{Q\} \triangleq (\forall S. P(S) \Rightarrow Q(\text{sem}(C, S)))$$

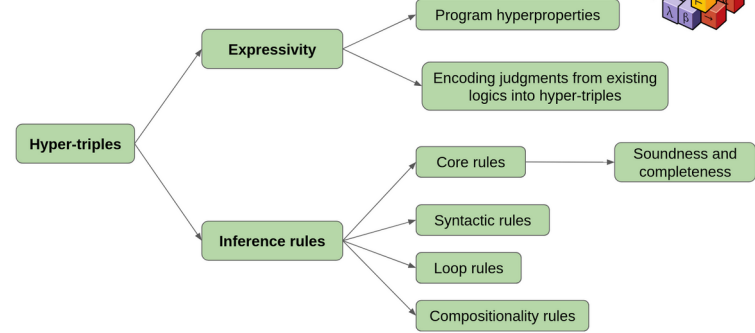
Hyper-assertions
(predicates on **sets** of states)

$$\text{sem}(C, S) \triangleq \{ \sigma' \mid \exists \sigma \in S. \langle C, \sigma \rangle \rightarrow \sigma' \}$$



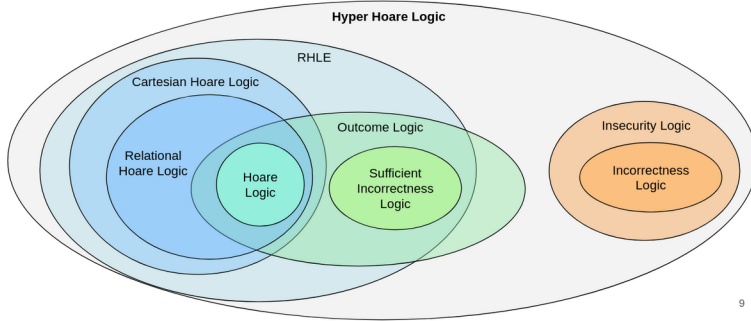
5

Contributions



Expressivity: Judgments from Existing Logics

expressivity of judgments for a **single non-deterministic IMP** program (no heap, no probabilities)



9

Future work

- Automation
- Extension to relational properties (more than one program)
- Extension to heap-manipulating programs