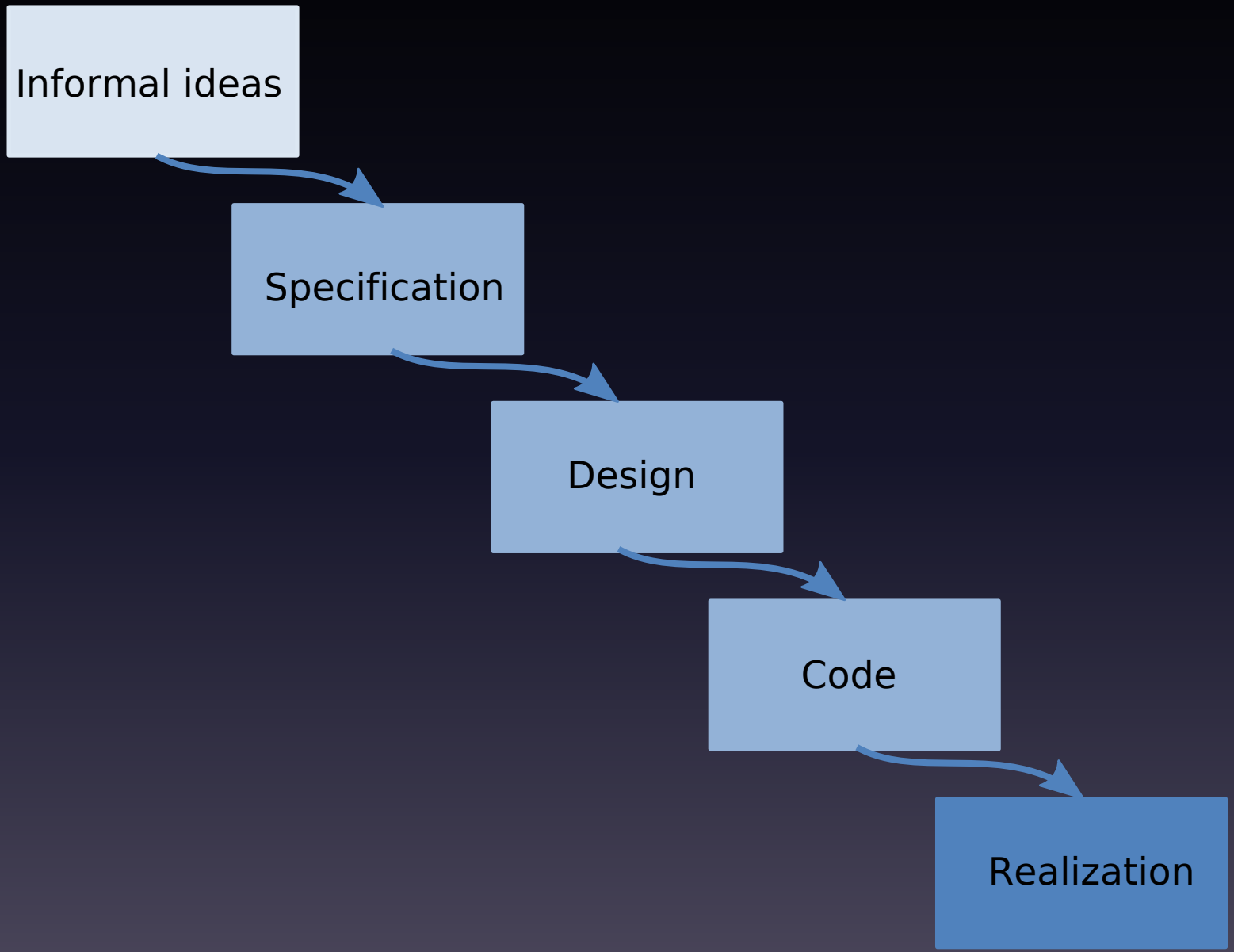
Complementary Criteria for Testing Temporal Logic Properties

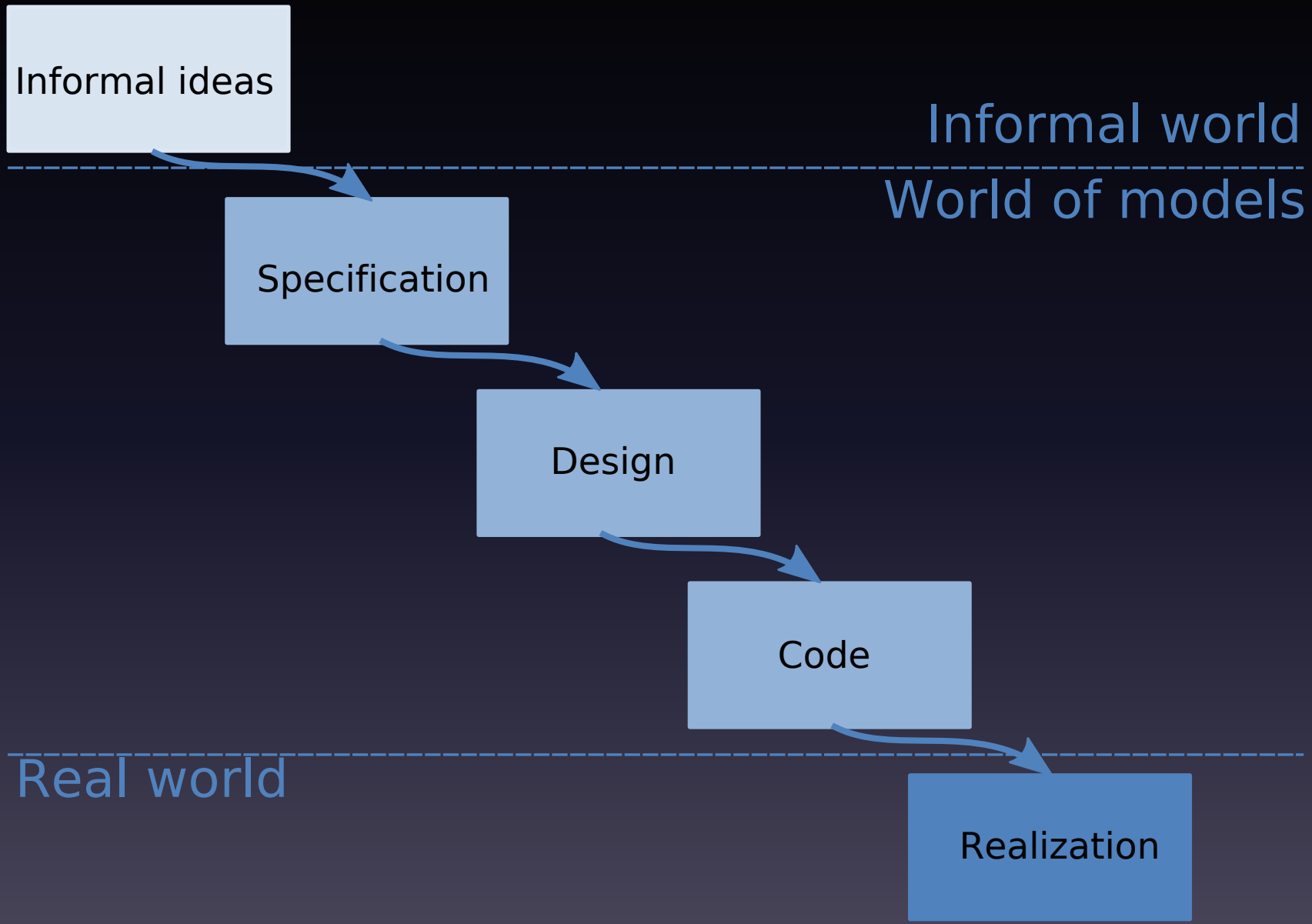
Gordon Fraser and Franz Wotawa

Graz University of Technology, Austria

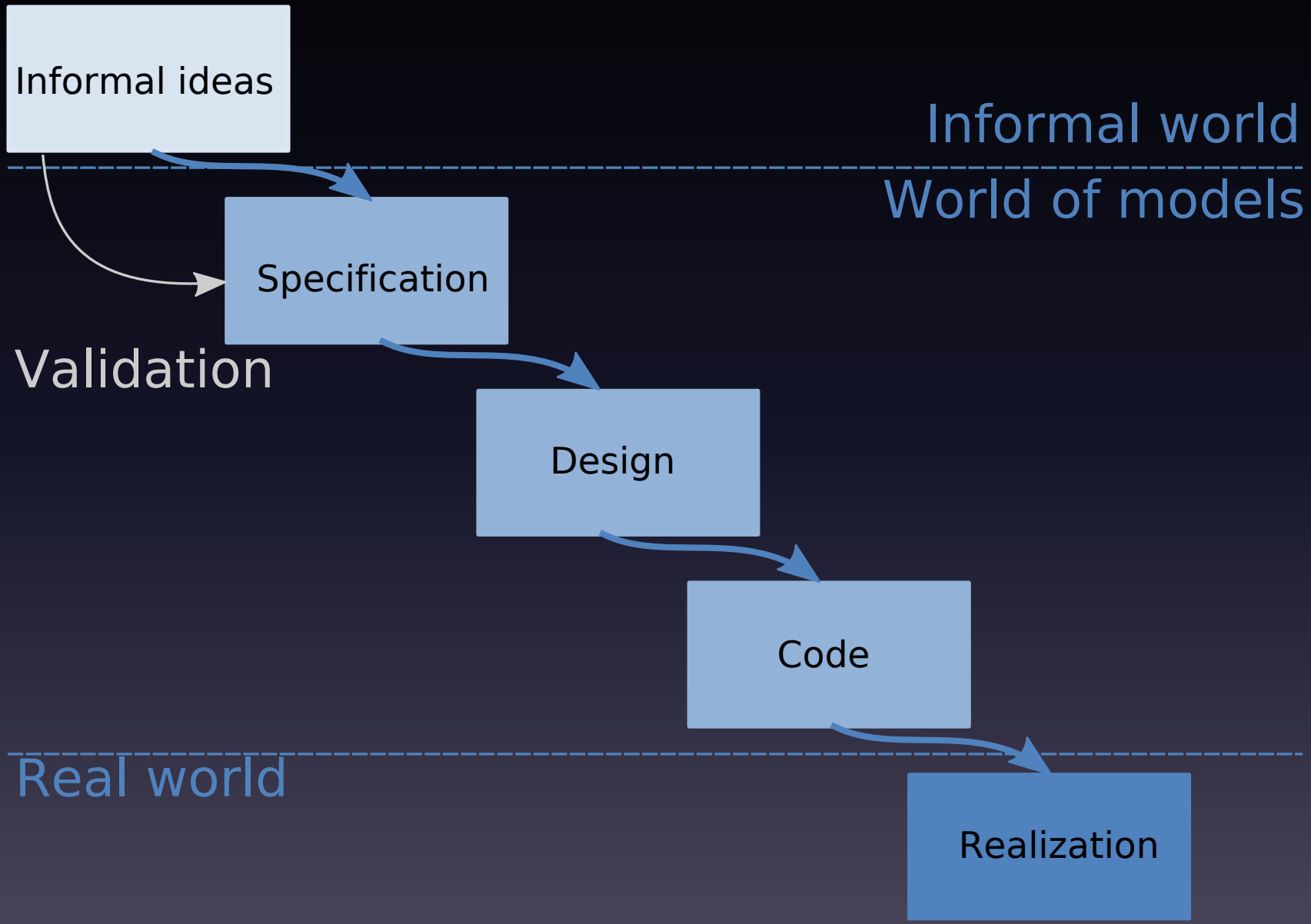
Model-Based Development Process



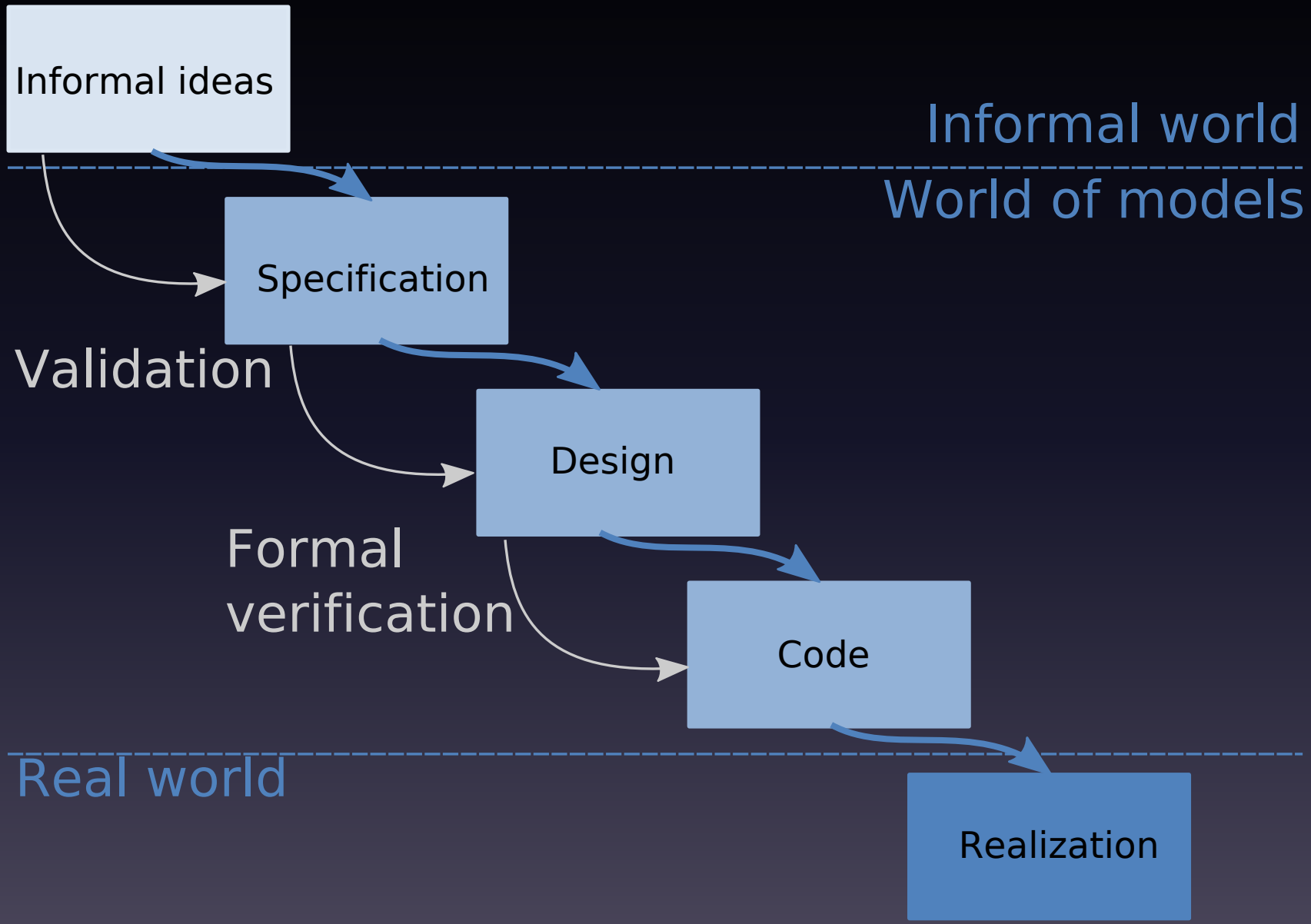
Model-Based Development Process



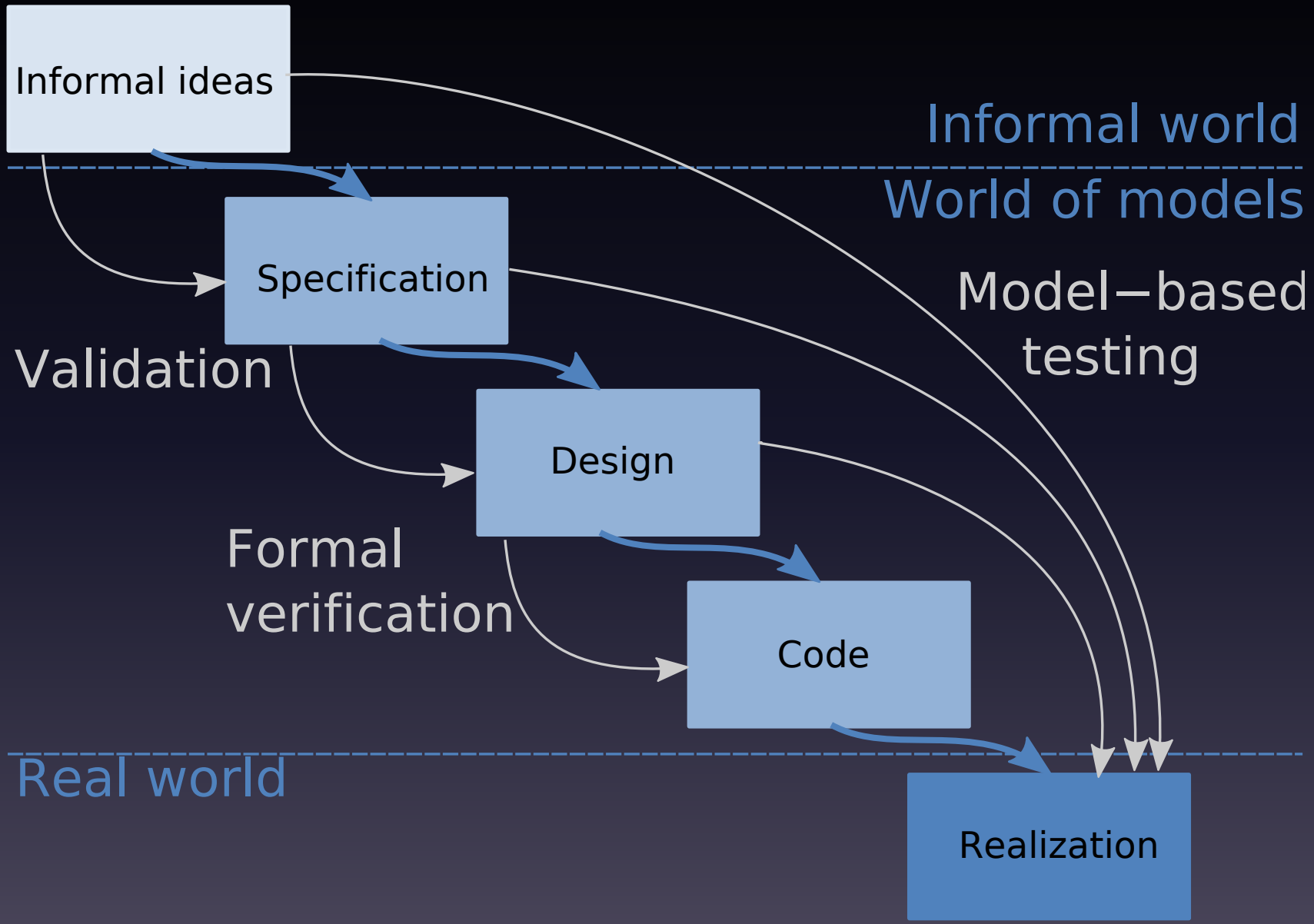
Model-Based Development Process



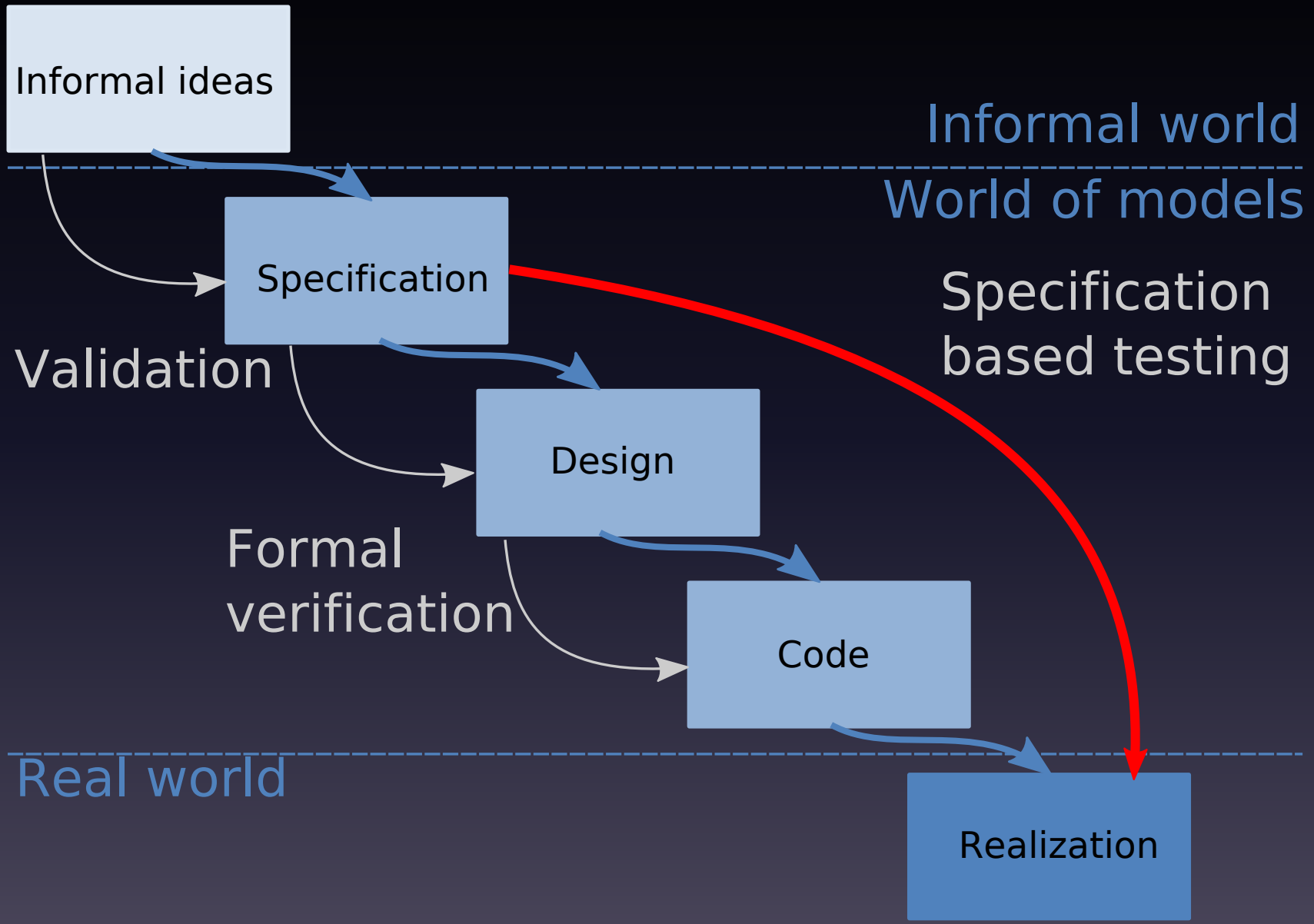
Model-Based Development Process



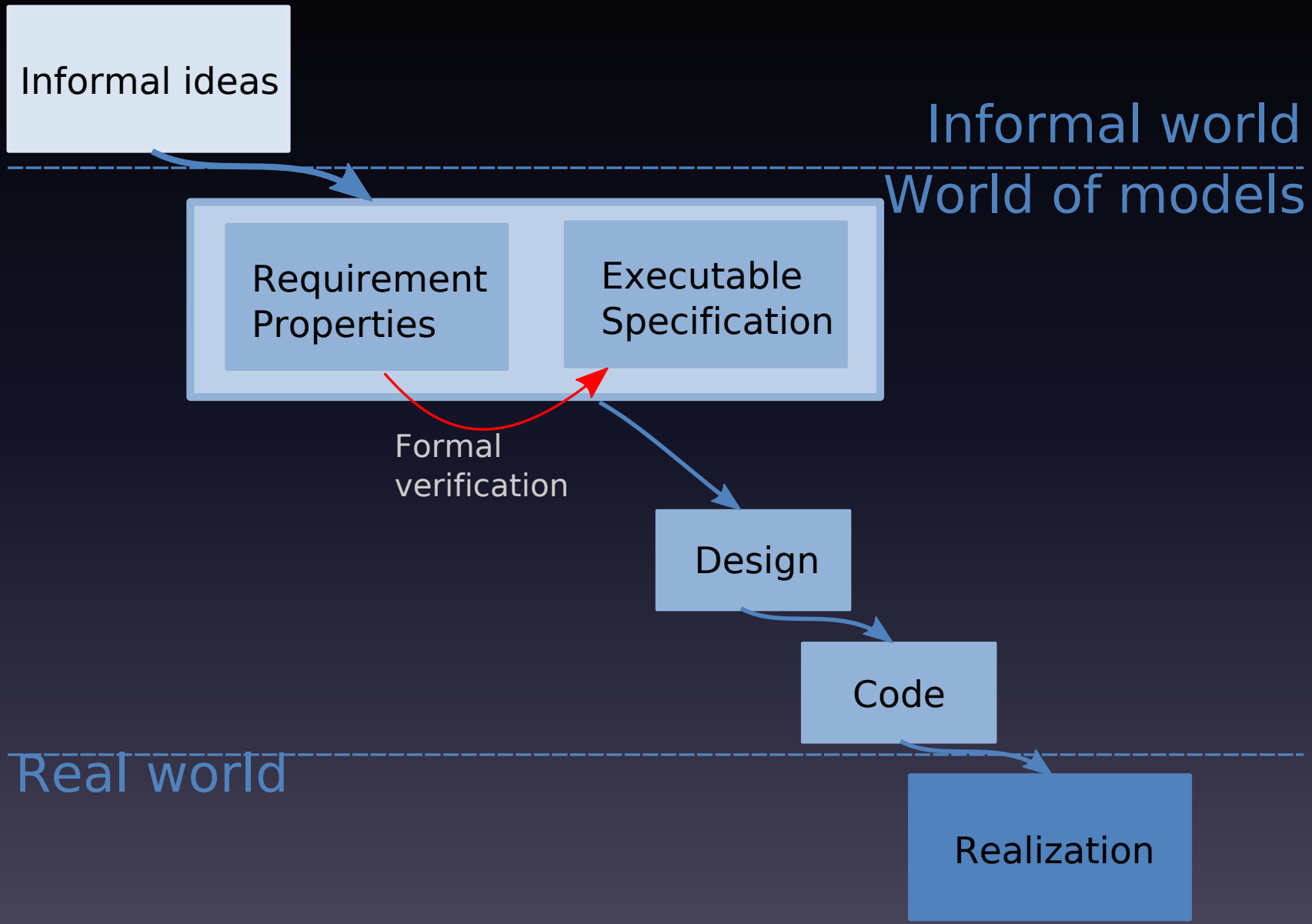
Model-Based Development Process



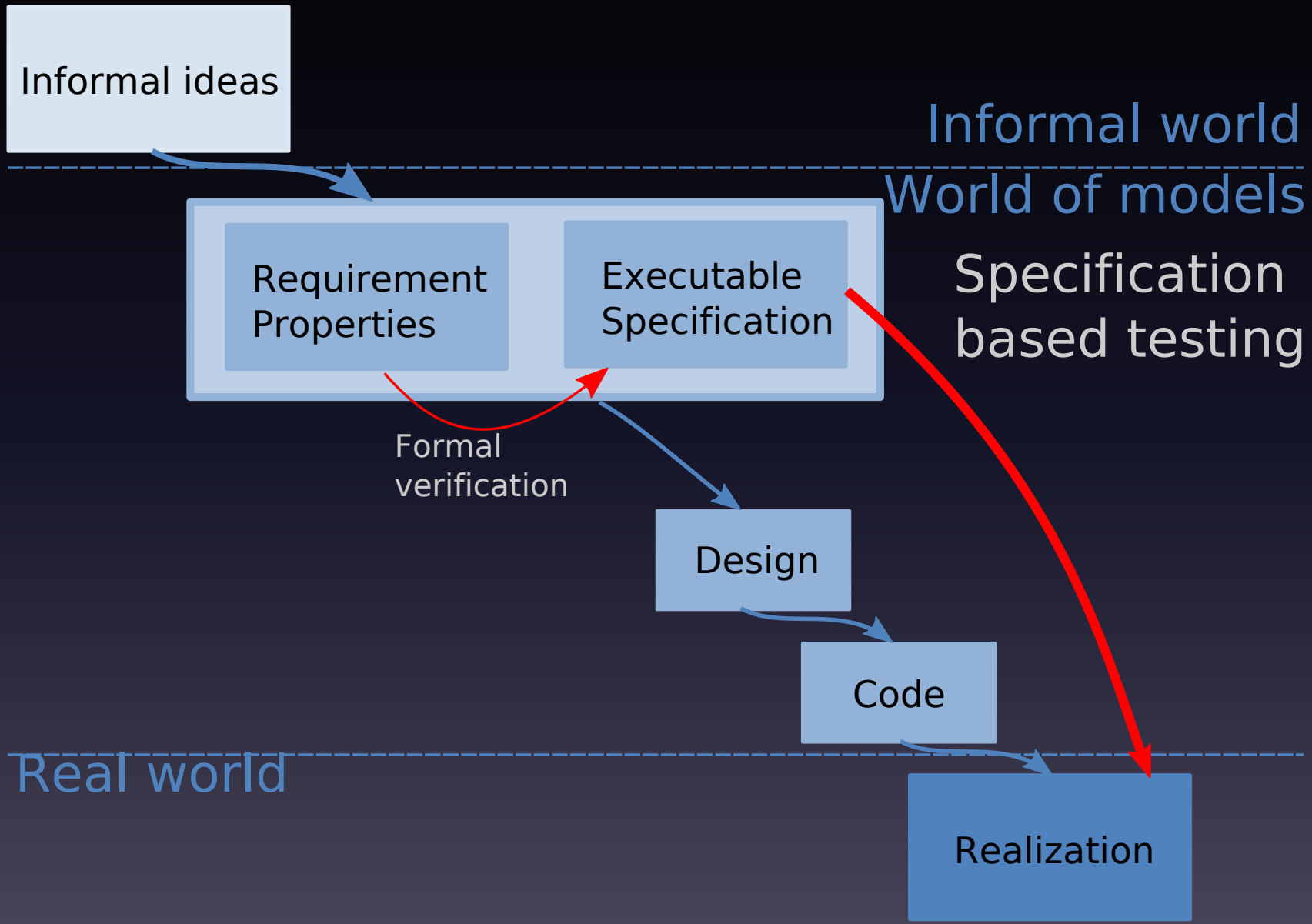
Specification Based Testing



Specification Based Testing



Specification Based Testing



Test cases derived from executable spec

- When is a property covered?
- How is it covered?
- How to generate tests for properties?
- Which tests to generate?

Test cases derived from executable spec

- When is a property covered?
- How is it covered?
- How to generate tests for properties?
- Which tests to generate?

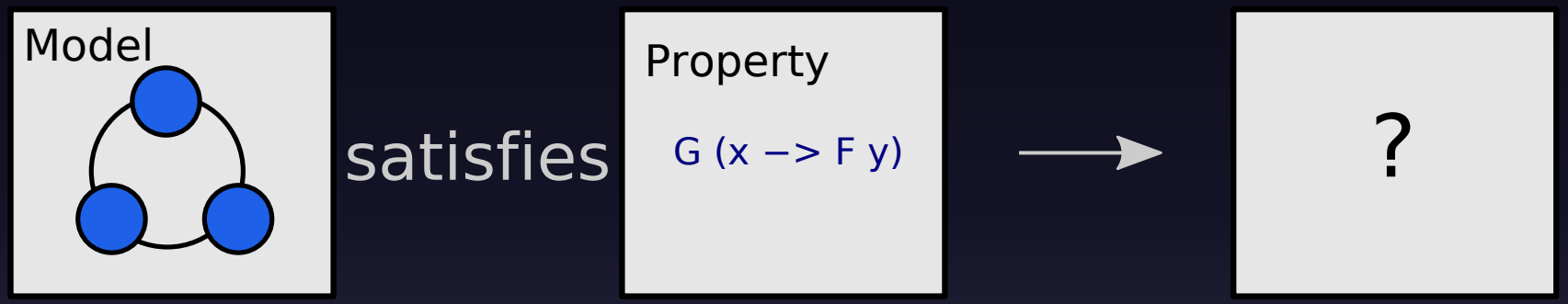
- Contribution: 2 new criteria to measure and generate tests

Test cases derived from executable spec

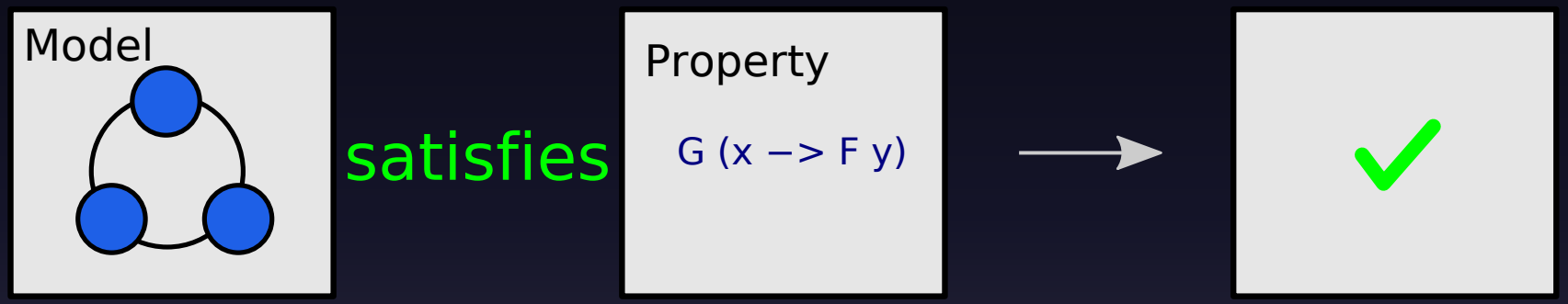
- When is a property covered?
- How is it covered?
- How to generate tests for properties?
- Which tests to generate?

- Contribution: 2 new criteria to measure and generate tests
- Assumption: Properties specified in temporal logic

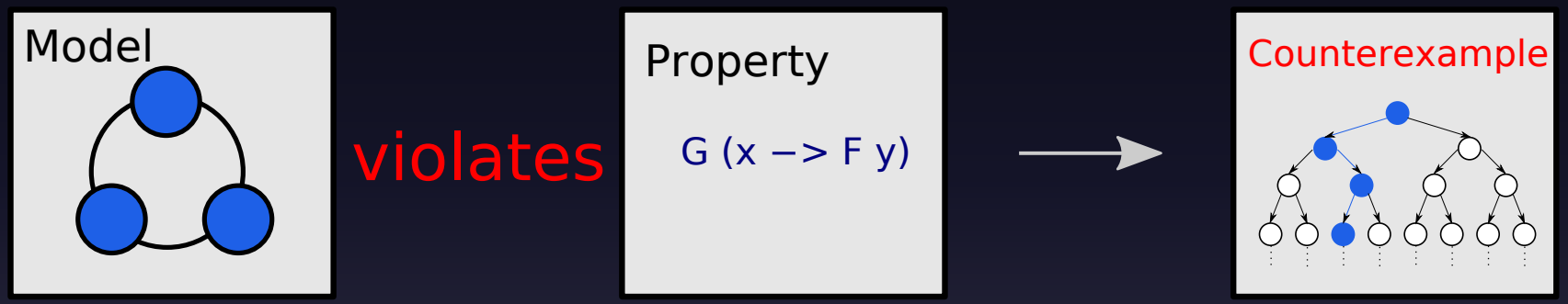
Model Checking



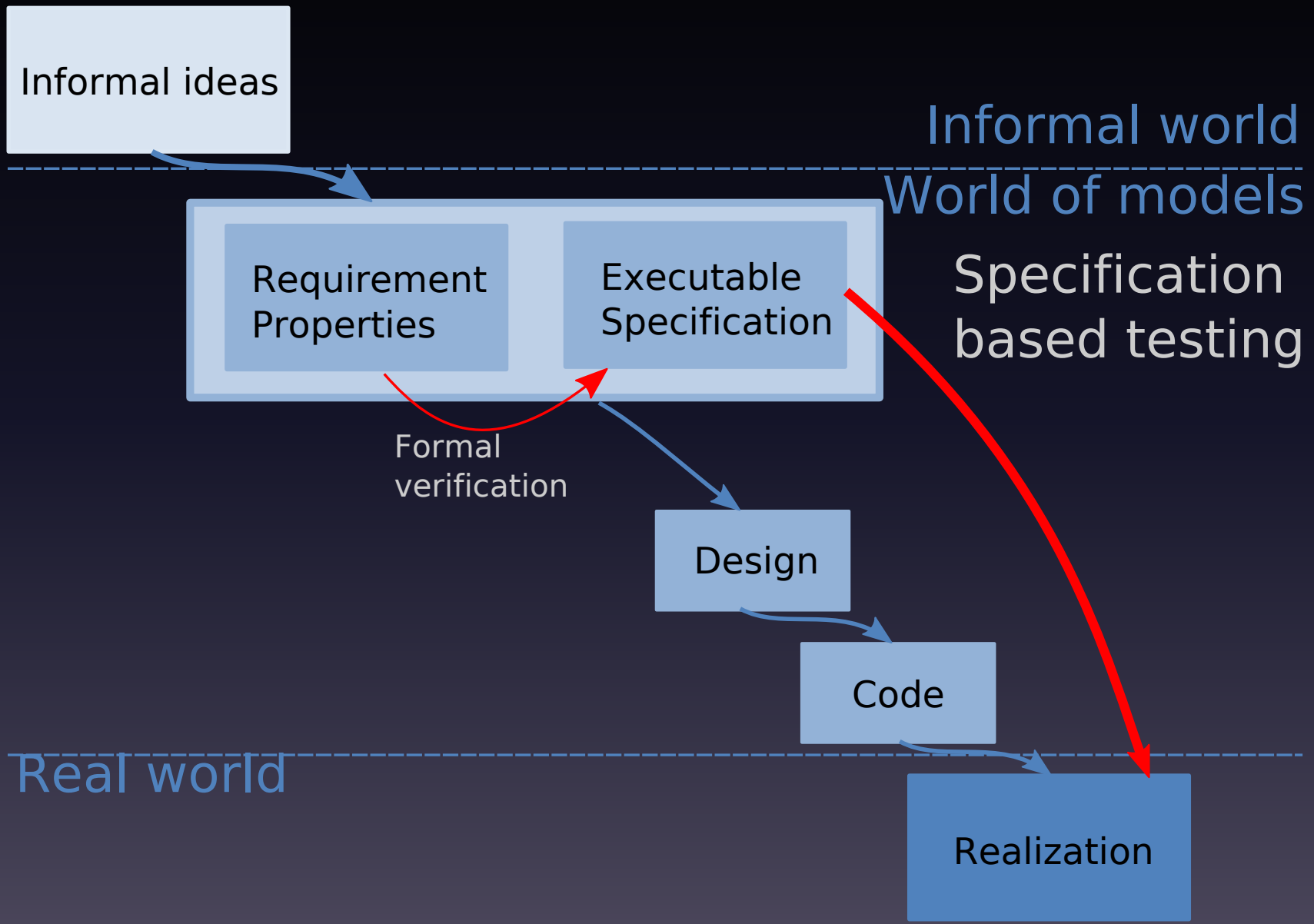
Model Checking



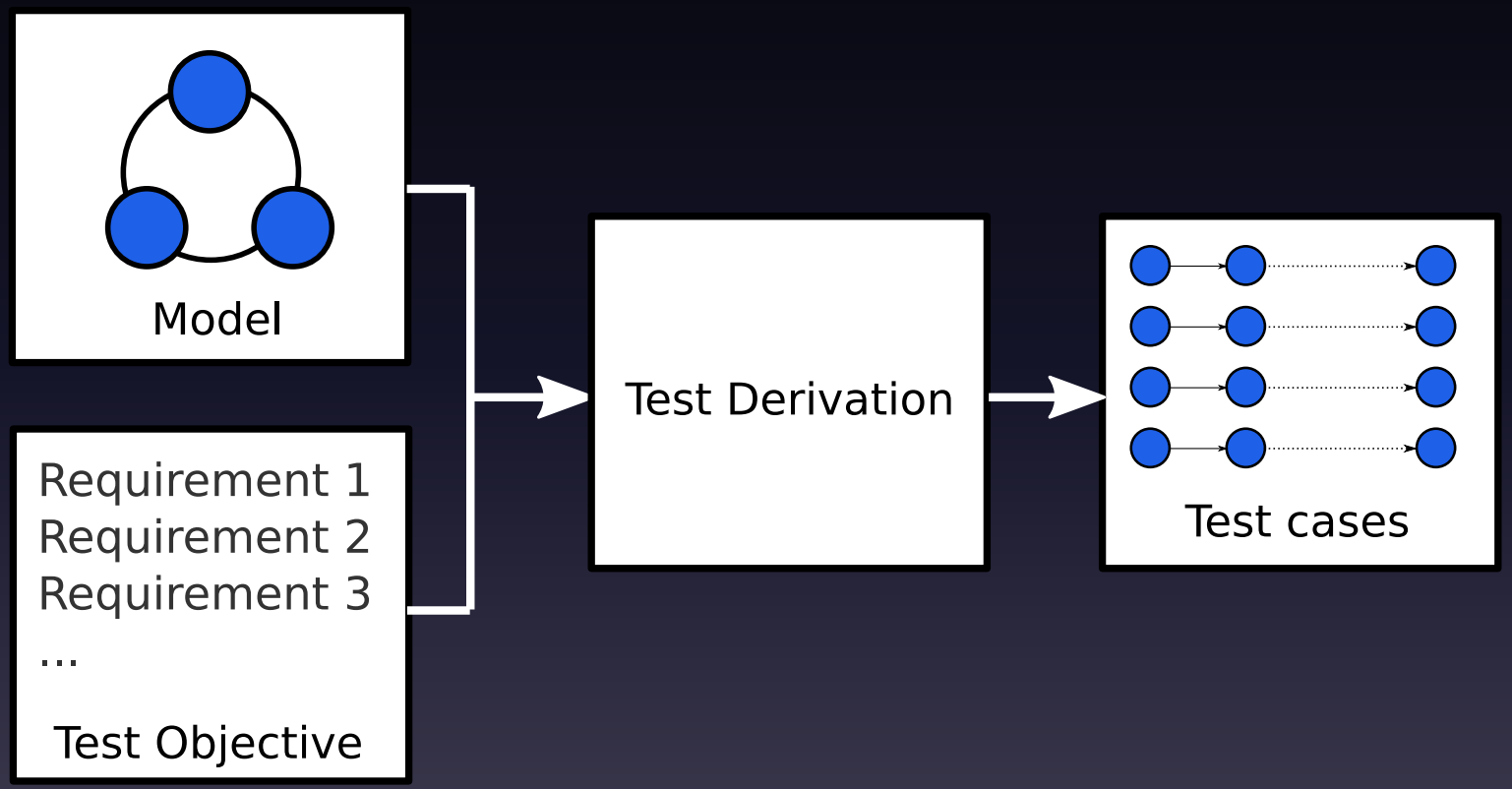
Model Checking



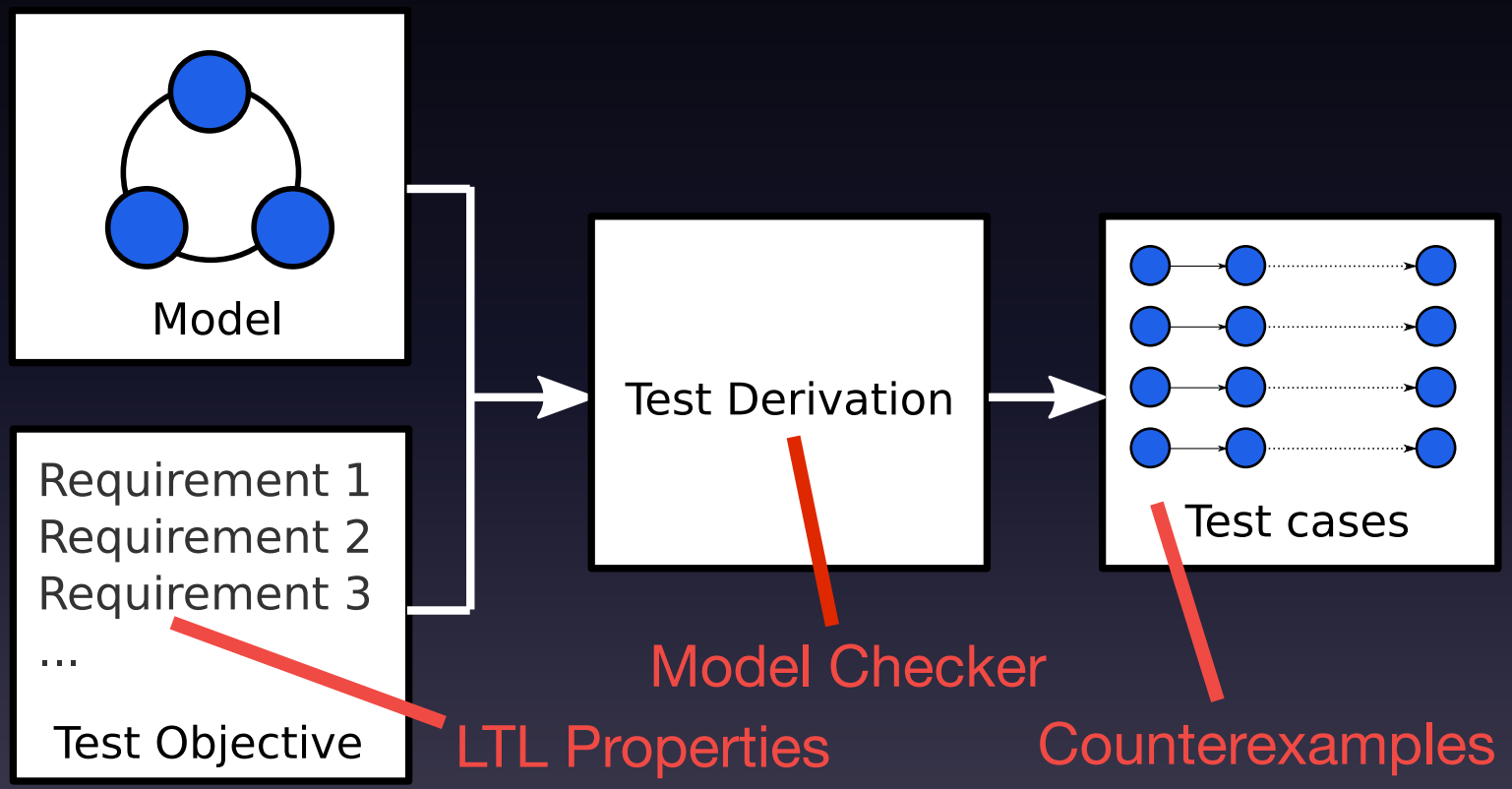
Specification Based Testing



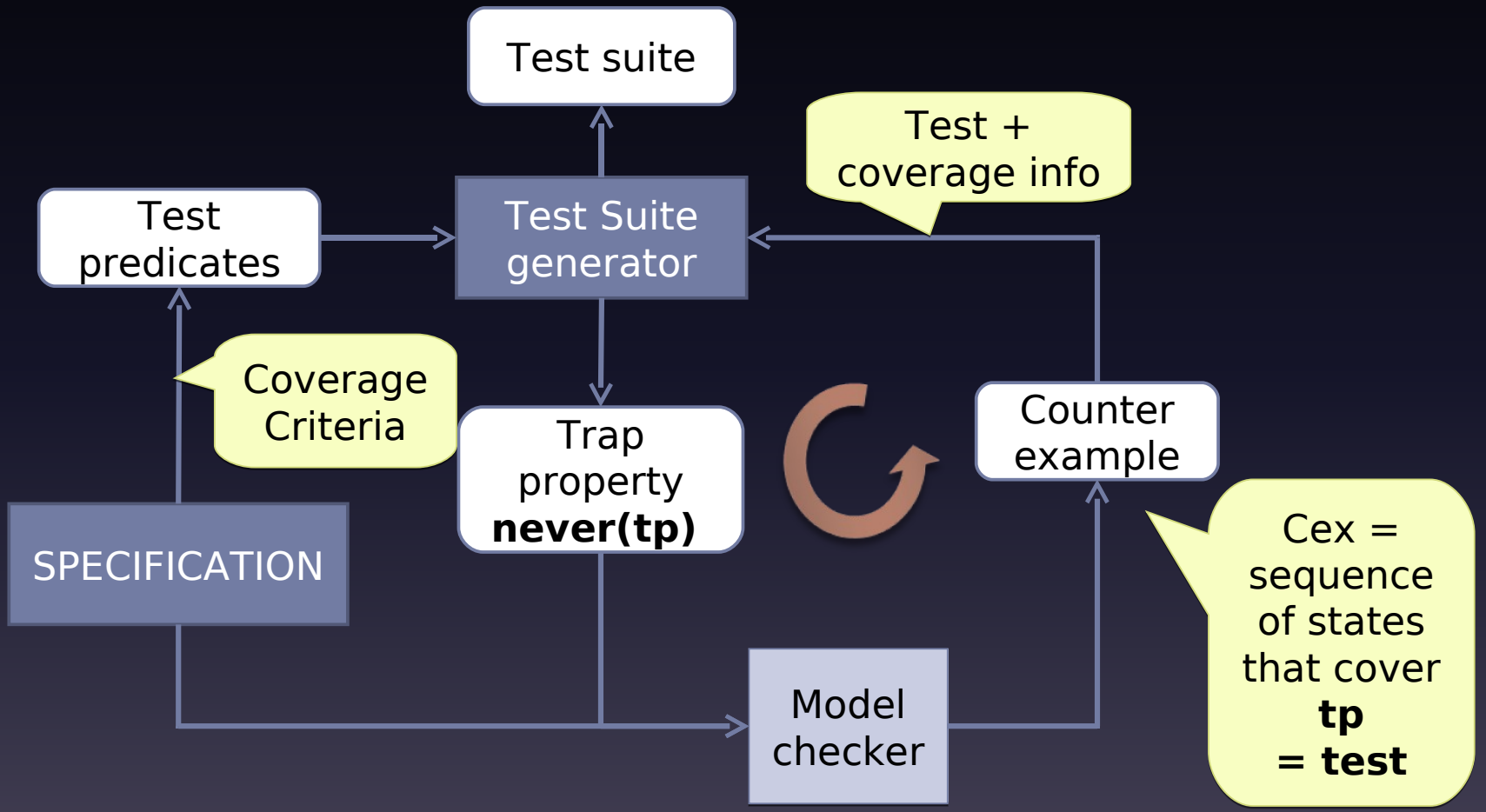
Specification Based Testing



Specification Based Testing



Test Case Generation with Model Checkers

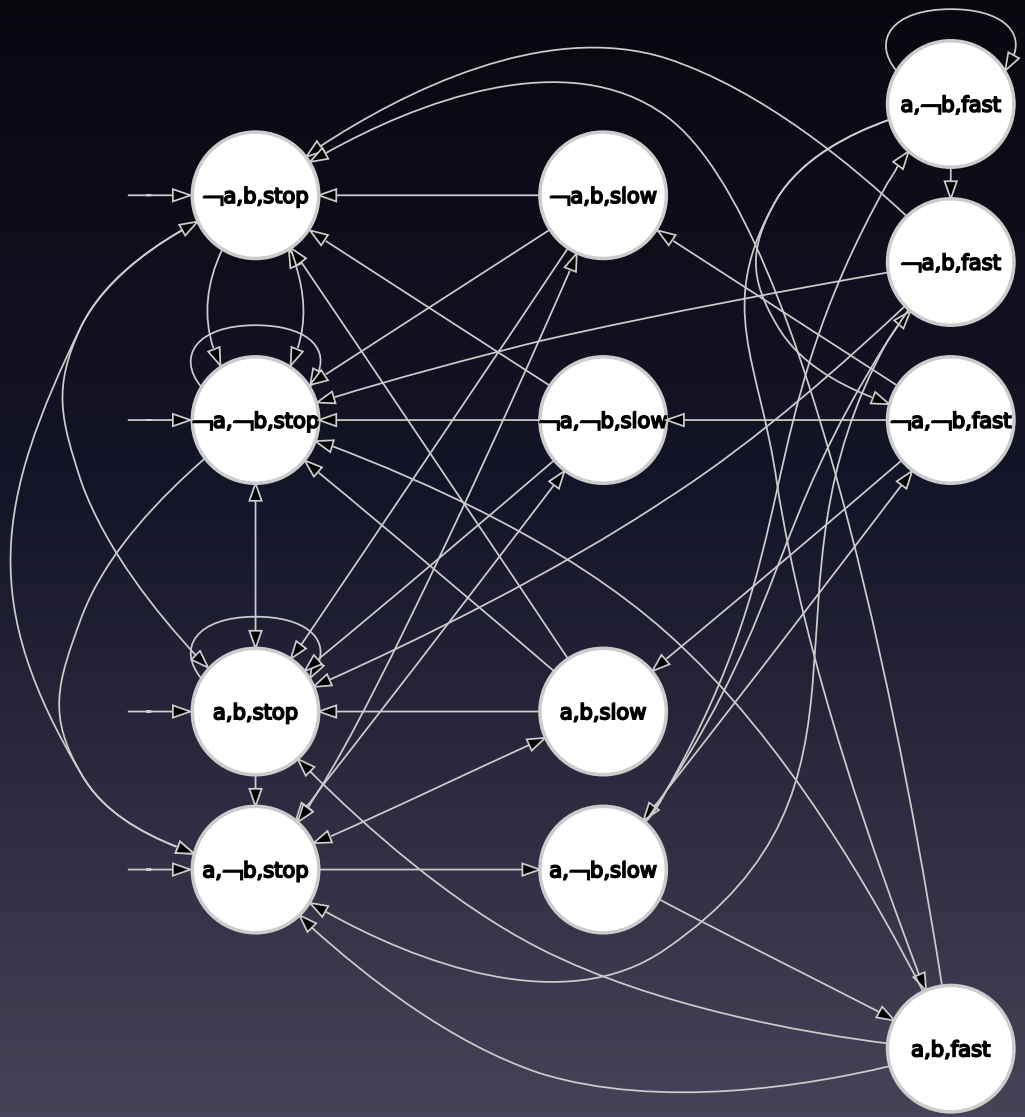


Example Specification: NuSMV

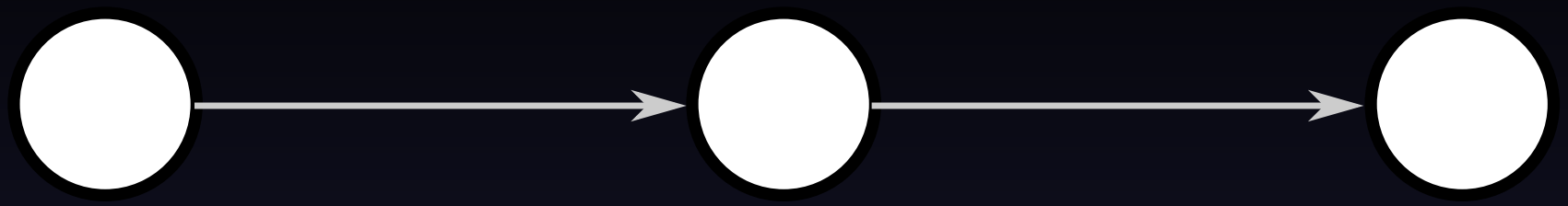
```
MODULE main
VAR
  accelerate: boolean;
  brake: boolean;
  velocity: { stop, slow, fast };

ASSIGN
  init(velocity) := stop;
  next(velocity) := case
     accelerate  & !brake & velocity = stop : slow;
     accelerate  & !brake & velocity = slow : fast;
     !accelerate & !brake & velocity = fast : slow;
     !accelerate & !brake & velocity = slow : stop;
     brake: stop;
     TRUE : velocity;
  esac;
```

Example Specification: NuSMV



Temporal Logics

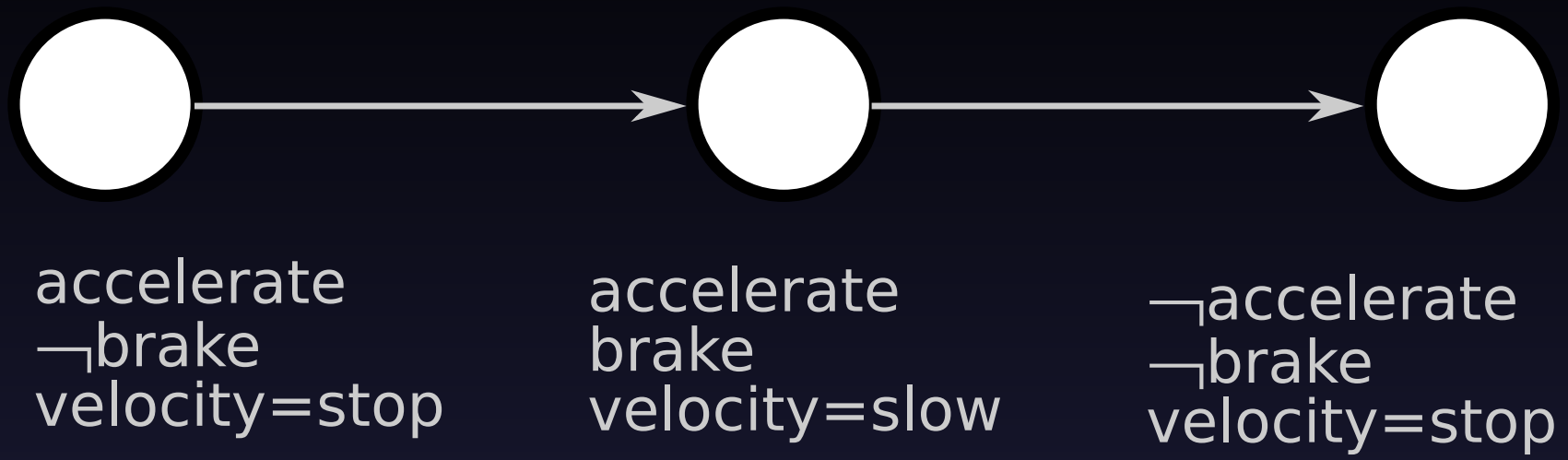


accelerate
¬brake
velocity=stop

accelerate
brake
velocity=slow

¬accelerate
¬brake
velocity=stop

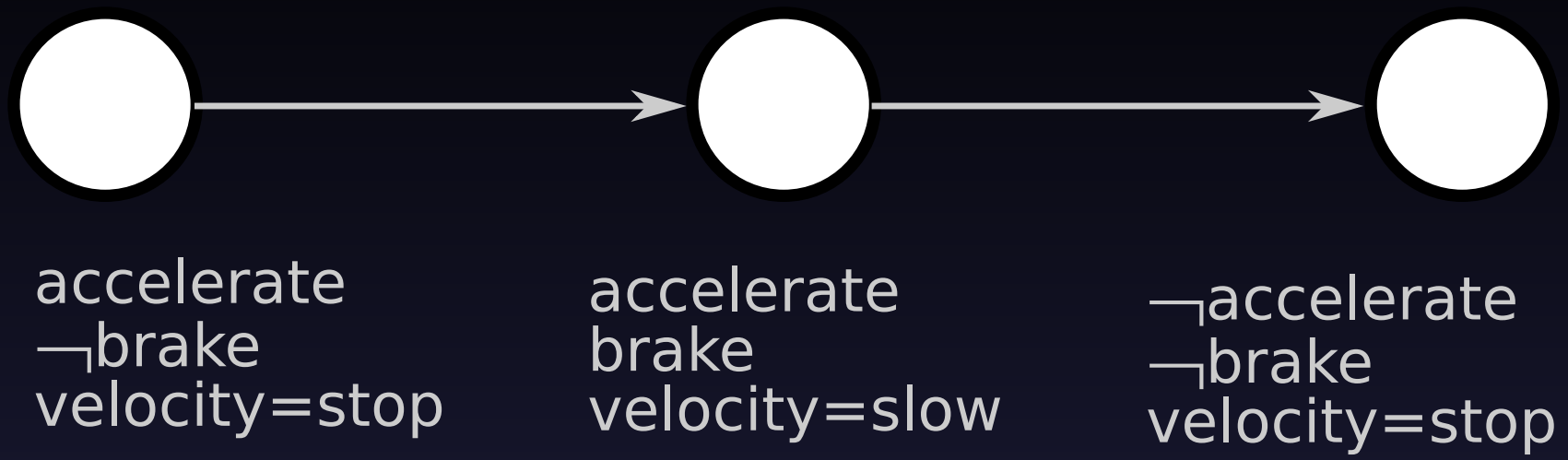
Temporal Logics



Example Property

$$\mathbf{G}\,\neg(\,\text{velocity} = \text{fast})$$

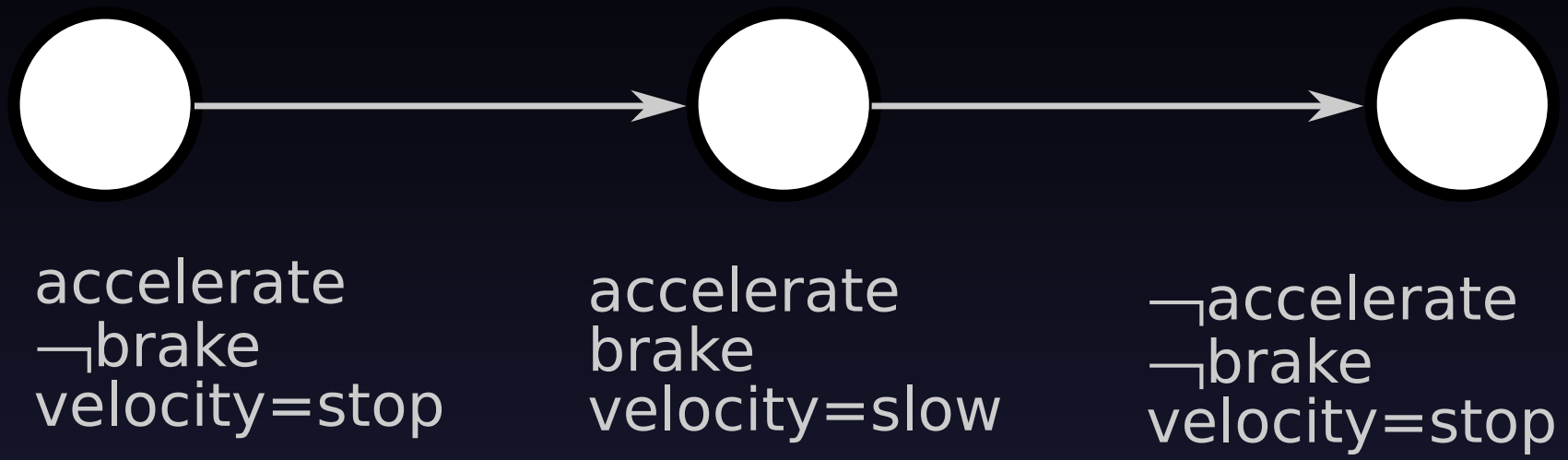
Temporal Logics



Example Property

X velocity = slow

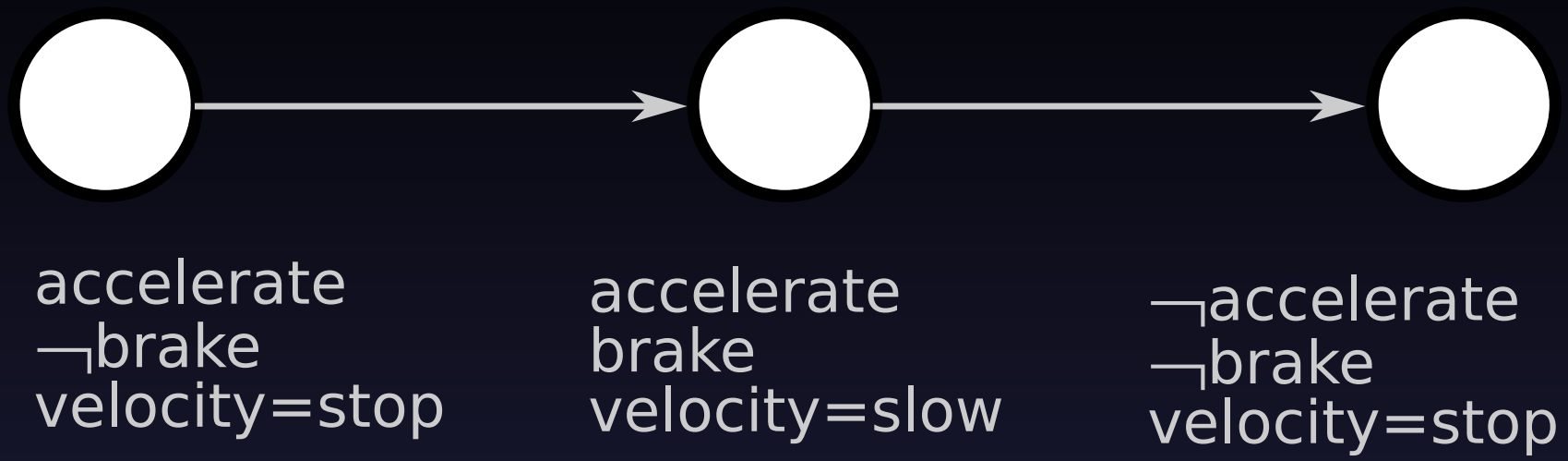
Temporal Logics



Example Property

F ¬ accelerate

Temporal Logics



Example Property

accelerate **U** brake

Example: NuSMV

```
MODULE main
VAR
  accelerate: boolean;
  brake: boolean;
  velocity: { stop, slow, fast };

ASSIGN
  init(velocity) := stop;
  next(velocity) := case
     accelerate  & !brake & velocity = stop : slow;
     accelerate  & !brake & velocity = slow : fast;
     !accelerate & !brake & velocity = fast : slow;
     !accelerate & !brake & velocity = slow : stop;
     brake: stop;
     TRUE : velocity;
  esac;
```

Example: NuSMV

```
MODULE main
VAR
  accelerate: boolean;
  brake: boolean;
  velocity: { stop, slow, fast };

ASSIGN
  init(velocity) := stop;
  next(velocity) := case
     accelerate  & !brake & velocity = stop : slow;
     accelerate  & !brake & velocity = slow : fast;
     !accelerate & !brake & velocity = fast : slow;
     !accelerate & !brake & velocity = slow : stop;
     brake: stop;
     TRUE : velocity;
  esac;
```

Coverage Criteria

```
accelerate  & !brake & velocity = stop : slow;
```

Predicate Coverage

Coverage Criteria

```
accelerate  & !brake & velocity = stop : slow;
```

Predicate Coverage

- $\mathbf{G}($ accelerate $\wedge \neg$ brake $\wedge$ velocity $=$ stop $\rightarrow$ $\mathbf{X} \neg($ velocity $=$ slow$))$

Coverage Criteria

```
accelerate  & !brake & velocity = stop : slow;
```

Predicate Coverage

- $\mathbf{G}(\ \text{accelerate} \wedge \neg\ \text{brake} \wedge\ \text{velocity} =\ \text{stop} \rightarrow \mathbf{X} \neg(\ \text{velocity} =\ \text{slow}))$
- $\mathbf{G}(\neg(\ \text{accelerate} \wedge \neg\ \text{brake} \wedge\ \text{velocity} =\ \text{stop}) \rightarrow \mathbf{X}(\ \text{velocity} =\ \text{slow}))$

Coverage Criteria

```
accelerate  & !brake & velocity = stop : slow;
```

Predicate Coverage

- $\mathbf{G}(\text{ accelerate} \wedge \neg \text{ brake} \wedge \text{ velocity} = \text{ stop} \rightarrow \mathbf{X} \neg(\text{ velocity} = \text{ slow}))$
- $\mathbf{G}(\neg(\text{ accelerate} \wedge \neg \text{ brake} \wedge \text{ velocity} = \text{ stop}) \rightarrow \mathbf{X}(\text{ velocity} = \text{ slow}))$

- 15 specification based test criteria
- 2 property based criteria
  1. Based on vacuity
  2. Based on MCDC

Unique First Cause Coverage

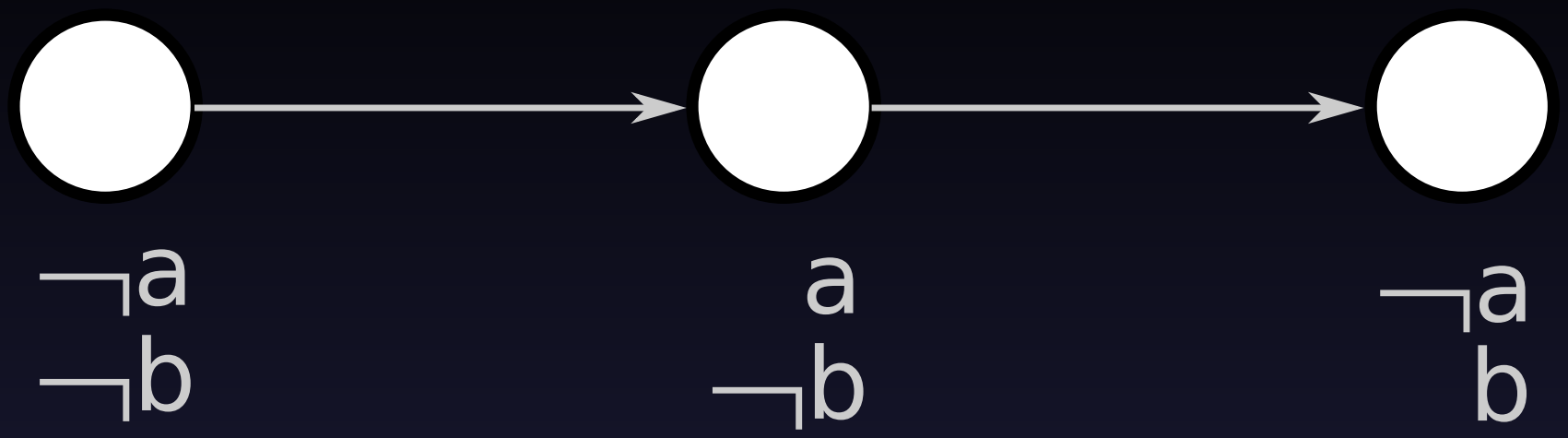
- MCDC: Test cases where clauses affect predicates

Unique First Cause Coverage

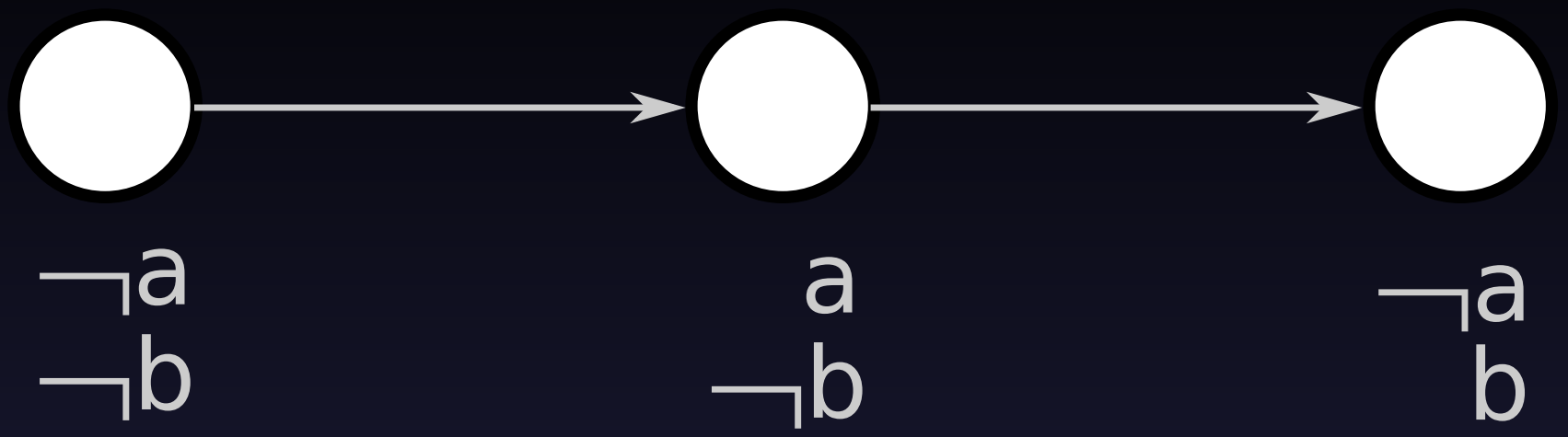
- MCDC: Test cases where clauses affect predicates
- UFC [Whalen et al., 2006]:

  Clause c is the unique first cause of a formula A, if in the first state along a path where A is satisfied, it is satisfied *because* of c

Unique First Cause Coverage



Unique First Cause Coverage



$$\mathbf{F}\,(a \vee b)$$

Unique First Cause Coverage

- Defined as set of rules
- Apply rules to property $\rightarrow$ set of test predicates
- Rules applied to all properties
- Test predicates: Measure coverage and generate tests

Unique First Cause Coverage is not enough

$$\mathbf{G}\left((c_1 \wedge c_2) \leftrightarrow \mathbf{X}\, d\right)$$

Unique First Cause Coverage is not enough

$$\mathbf{G}\left((c_1 \wedge c_2) \leftrightarrow \mathbf{X}\, d\right)$$

- UFC covers:
  - Where c_1 causes d to be false.
  - Where c_2 causes d to be false.
  - Where $c_1 \wedge c_2$ causes d to be true.

Unique First Cause Coverage is not enough

$$\mathbf{G}\left((c_1 \wedge c_2) \leftrightarrow \mathbf{X}\, d\right)$$

- UFC covers:
  - Where c_1 causes d to be false.
  - Where c_2 causes d to be false.
  - Where $c_1 \wedge c_2$ causes d to be true.
- What if d represents a safety critical value?

Unique First Cause Coverage is not enough

$$\mathbf{G}\left((c_1 \wedge c_2) \leftrightarrow \mathbf{X}\, d\right)$$

- UFC covers:
  - Where c_1 causes d to be false.
  - Where c_2 causes d to be false.
  - Where $c_1 \wedge c_2$ causes d to be true.
- What if d represents a safety critical value?
- Is $\neg c_1 \wedge \neg c_2$ covered?

Unique First Cause Coverage is not enough

$$\mathbf{G}\left((c_1 \wedge c_2) \leftrightarrow \mathbf{X}\, d\right)$$

- UFC covers:
  - Where c_1 causes d to be false.
  - Where c_2 causes d to be false.
  - Where $c_1 \wedge c_2$ causes d to be true.
- What if d represents a safety critical value?
- Is $\neg c_1 \wedge \neg c_2$ covered?
- $\rightarrow$ Property Inactive Clause Coverage

Property Inactive Clause Coverage

$$
\begin{aligned}
\rho(x) &= \{x, \neg x\} \\
\rho(A \wedge B) &= \{a \wedge \neg B \mid a \in \rho(A)\} \cup \{\neg A \wedge b \mid b \in \rho(B)\} \\
\rho(A \vee B) &= \{a \wedge B \mid a \in \rho(A)\} \cup \{A \wedge b \mid b \in \rho(B)\} \\
\rho(\neg A) &= \rho(A) \\
\rho(\mathbf{G}(A)) &= \{A \,\mathbf{U}\, (a \wedge \mathbf{G}(A)) \mid a \in \rho(A)\} \\
\rho(\mathbf{F}(A)) &= \{\neg A \,\mathbf{U}\, (a \wedge (A \vee \mathbf{F}\, A)) \mid a \in \rho(A)\} \\
\rho(\mathbf{X}(A)) &= \{\mathbf{X}(a) \mid a \in \rho(A)\} \\
\rho(A \,\mathbf{U}\, B) &= \{(A \wedge \neg B) \,\mathbf{U}\, (a \wedge (B \vee (A \,\mathbf{U}\, B))) \mid a \in \rho(A)\} \cup \\
&\quad \{(A \wedge \neg B) \,\mathbf{U}\, (b \wedge (B \vee (A \,\mathbf{U}\, B))) \mid b \in \rho(B)\}
\end{aligned}
$$

Property Inactive Clause Coverage

Property

$\phi = \mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$

Property Inactive Clause Coverage

Property

$$\phi = \mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$$

Rule

$$\rho(\mathbf{G}\,(A)) = \{A\,\mathbf{U}\,(a \wedge \mathbf{G}\,(A)) \mid a \in \rho(A)\}$$

Property Inactive Clause Coverage

Property

$$\phi = \mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$$

Rule

$$\rho(\mathbf{G}\,(A)) = \{A\,\mathbf{U}\,(a \wedge \mathbf{G}\,(A)) \mid a \in \rho(A)\}$$

Result

$$(x \rightarrow \mathbf{X}\,y)\,\mathbf{U}\,(a \wedge \mathbf{G}\,(x \rightarrow \mathbf{X}\,y)) \mid a \in \rho(x \rightarrow \mathbf{X}\,y)$$

Property Inactive Clause Coverage

Property

$x \rightarrow \mathbf{X}\, y$

Property Inactive Clause Coverage

Property

$x \rightarrow \mathbf{X}\, y$

Rule

$\rho(A \vee B) = \{a \wedge B \mid a \in \rho(A)\} \cup \{A \wedge b \mid b \in \rho(B)\}$
$\rho(\neg A) = \rho(A)$

Property Inactive Clause Coverage

Property

$x \rightarrow \mathbf{X}\, y$

Rule

$\rho(A \vee B) = \{a \wedge B \mid a \in \rho(A)\} \cup \{A \wedge b \mid b \in \rho(B)\}$
$\rho(\neg A) = \rho(A)$

Result

$\rho(x \rightarrow \mathbf{X}\, y) = \rho(\neg x \vee \mathbf{X}\, y)$
$= \{a \wedge \mathbf{X}\, y \mid a \in \rho(\neg x)\} \cup \{\neg x \wedge b \mid b \in \rho(\mathbf{X}\, y)\}$

Property Inactive Clause Coverage

Property
$\mathbf{X}\, y$

Property Inactive Clause Coverage

Property

$\mathbf{X}\, y$

Rule

$\rho(\mathbf{X}\,(A)) = \{\, \mathbf{X}\,(a) \mid a \in \rho(A)\}$

Property Inactive Clause Coverage

Property

$\mathbf{X}\, y$

Rule

$\rho(\mathbf{X}\,(A)) = \{\mathbf{X}\,(a) \mid a \in \rho(A)\}$

Result

$\rho(\mathbf{X}\, y) = \{\mathbf{X}\,(a) \mid a \in \rho(y)\} = \{\mathbf{X}\,(y), \mathbf{X}\,(\neg y)\}$

Property Inactive Clause Coverage

Property

$$\phi = \mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$$

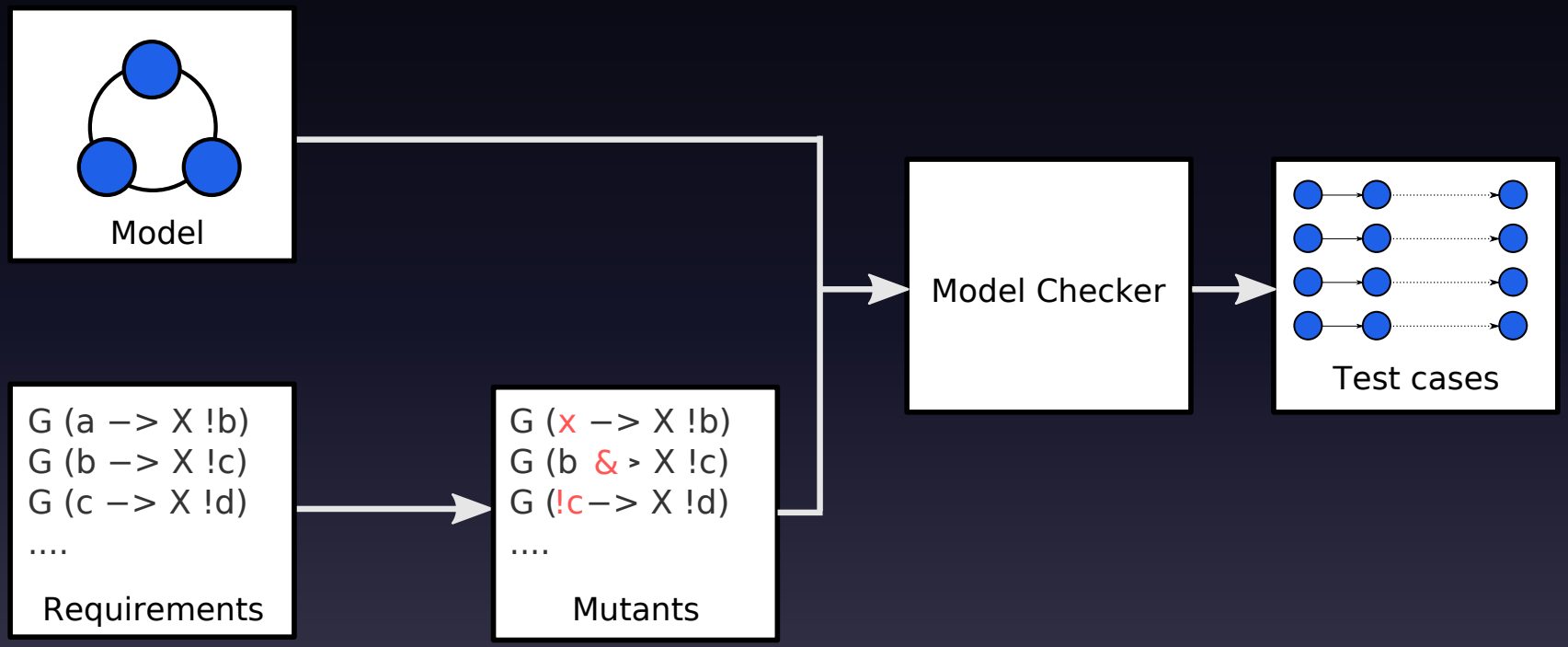
Result

$$\{(x \rightarrow \mathbf{X}\,y)\,\mathbf{U}\,((x \wedge \mathbf{X}\,y) \wedge \mathbf{G}\,(x \rightarrow \mathbf{X}\,y)),$$
$$(x \rightarrow \mathbf{X}\,y)\,\mathbf{U}\,((\neg x \wedge \mathbf{X}\,y) \wedge \mathbf{G}\,(x \rightarrow \mathbf{X}\,y)),$$
$$(x \rightarrow \mathbf{X}\,y)\,\mathbf{U}\,((\neg x \wedge \mathbf{X}\,(y)) \wedge \mathbf{G}\,(x \rightarrow \mathbf{X}\,y)),$$
$$(x \rightarrow \mathbf{X}\,y)\,\mathbf{U}\,((\neg x \wedge \mathbf{X}\,(\neg y)) \wedge \mathbf{G}\,(x \rightarrow \mathbf{X}\,y))\}$$

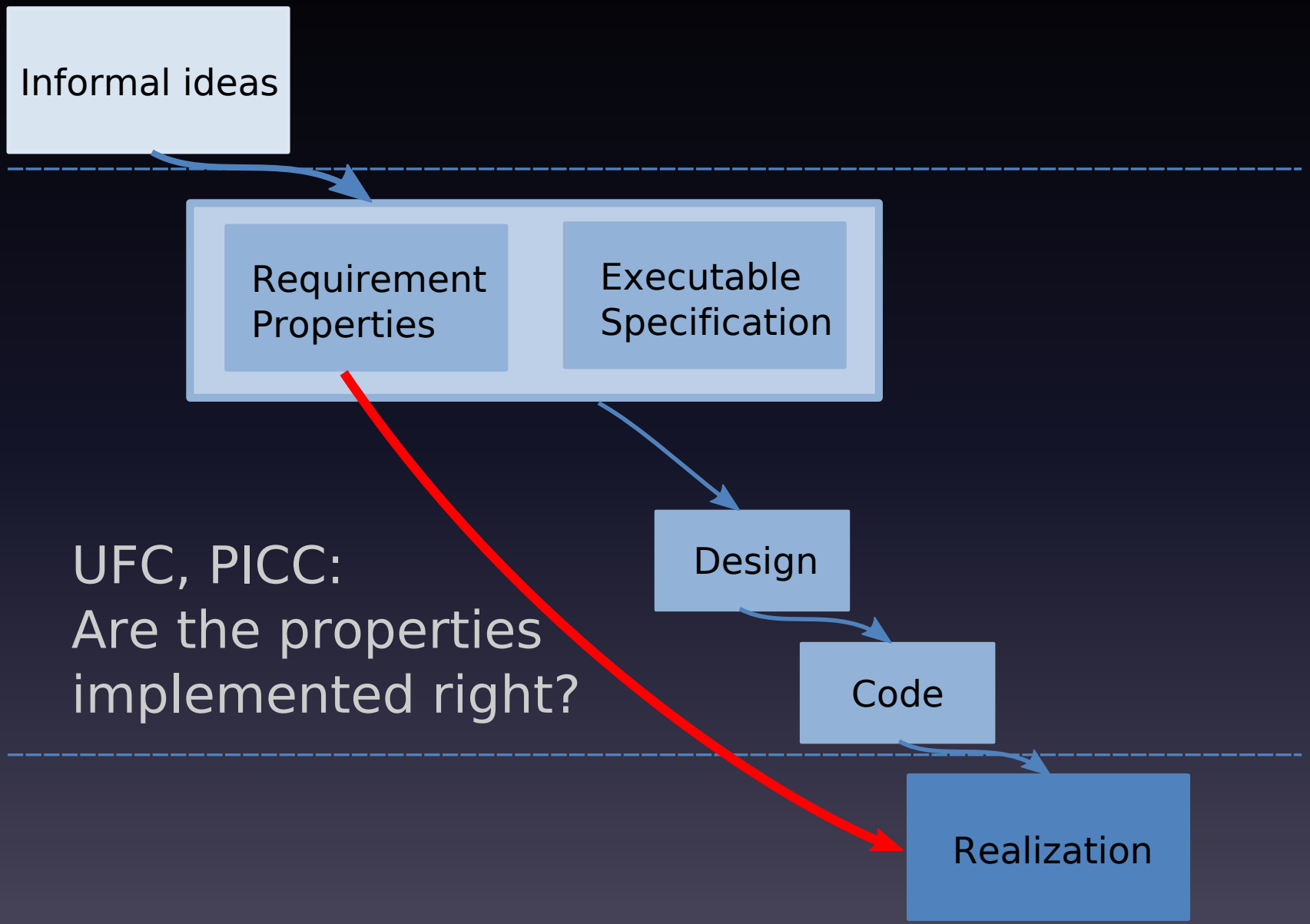
PICC Summary

- When do clauses *not* affect result?
- PICC extends UFC like RCDC extends MCDC
- Apply rules to all properties $\rightarrow$ test predicates

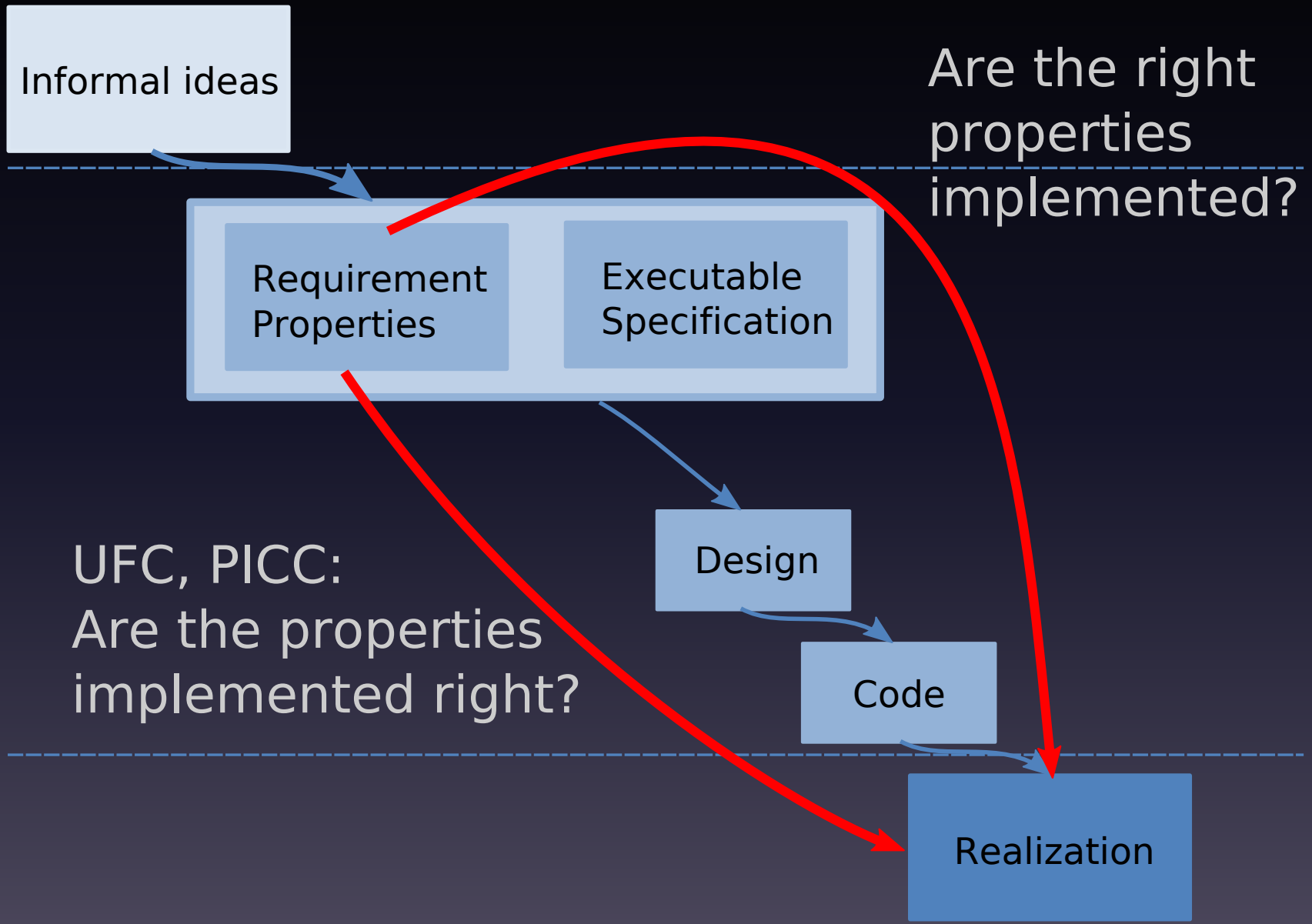
Mutation Testing for Properties



Mutation Testing for Properties



Mutation Testing for Properties



Mutation Testing for Properties

- Test whether wrong properties are *not* implemented
- Coupling effect
- If mutants are not implemented, complex wrong properties are not implemented either

Property Mutation

Example Property

$$\mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$$

Property Mutation

Example Property

$\mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$

Example mutants

- LRO: $\mathbf{G}\,(x \leftarrow \mathbf{X}\,y)$

Property Mutation

Example Property

$\mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$

Example mutants

- LRO: $\mathbf{G}\,(x \leftarrow \mathbf{X}\,y)$
- ORO: $\mathbf{G}\,(x \rightarrow \mathbf{X}\ x\ )$

Property Mutation

Example Property

$\mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$

Example mutants

- LRO: $\mathbf{G}\,(x \leftarrow \mathbf{X}\,y)$
- ORO: $\mathbf{G}\,(x \rightarrow \mathbf{X}\ x\ )$
- ...

Mutation Operators for LTL

$$\mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$$

Mutation Operators for LTL

$\mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$

Temporal Operator Insertion

$\mathbf{G}\,(\ \mathbf{X}\ x \rightarrow \mathbf{X}\,y)$

Mutation Operators for LTL

$\mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$

Temporal Operator Insertion

$\mathbf{G}\,(\ \mathbf{X}\ x \rightarrow \mathbf{X}\,y)$

Temporal Operator Replacement

$\mathbf{G}\,(x \rightarrow \mathbf{G}\ y)$

Mutation Operators for LTL

$\mathbf{G}\,(x \rightarrow \mathbf{X}\,y)$

Temporal Operator Insertion

$\mathbf{G}\,(\ \mathbf{X}\ x \rightarrow \mathbf{X}\,y)$

Temporal Operator Replacement

$\mathbf{G}\,(x \rightarrow \mathbf{G}\ y)$

Missing Temporal Operator

$\mathbf{G}\,(x \rightarrow y)$

Property Mutation Summary

- A mutant of a property is a trap property
- Each property results in several mutants for each mutation operator
- Test case generation: Create mutants for all properties
- Property mutation measures sensitivity wrt implemented properties

Case Study

- PICC and Mutation add new, previously uncovered test predicates

Case Study

- PICC and Mutation add new, previously uncovered test predicates
- 25% infeasible test predicates average
- 45% equivalent property mutants

Case Study

- PICC and Mutation add new, previously uncovered test predicates
- 25% infeasible test predicates average
- 45% equivalent property mutants
- 16% of literals are vacuously satisfied
- Mutation operators can be optimized to reduce equivalent mutants

Summary

- Test in addition to verification

Summary

- Test in addition to verification
- Derive tests from properties in addition to specifications

Summary

- Test in addition to verification
- Derive tests from properties in addition to specifications
- Test inactive clauses in addition to active clauses

Summary

- Test in addition to verification
- Derive tests from properties in addition to specifications
- Test inactive clauses in addition to active clauses
- Test that wrong properties are not implemented, in addition to testing that right properties are implemented

Summary

- Test in addition to verification
- Derive tests from properties in addition to specifications
- Test inactive clauses in addition to active clauses
- Test that wrong properties are not implemented, in addition to testing that right properties are implemented

- Thank you for your attention!
- Questions?