22c181:
Formal Methods in Software Engineering

The University of Iowa

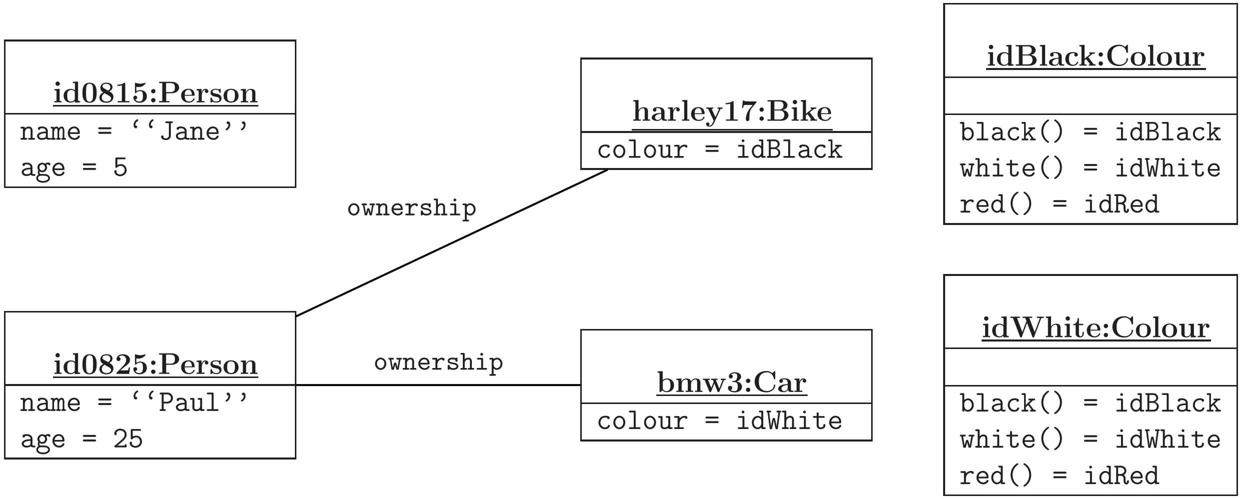
Spring 2008

From OCL to Propositional and First-order Logic: Part I

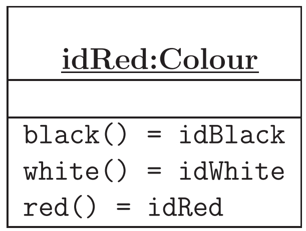
Contents

- Overview of KeY
- UML and its semantics
- Introduction to OCL
- Specifying requirements with OCL
- Modelling of Systems with Formal Semantics
- Propositional & First-order logic, sequent calculus
- OCL to Logic, horizontal proof obligations, using KeY
- Dynamic logic, proving program correctness
- Java Card DL
- Vertical proof obligations, using KeY
- Wrap-up, trends

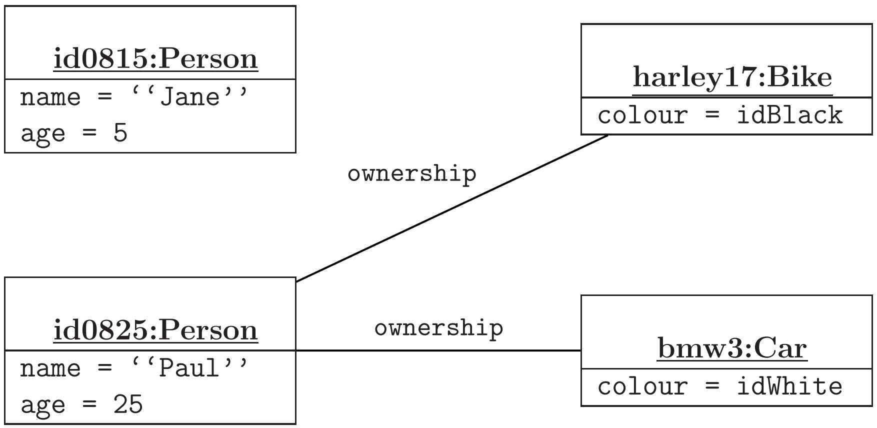
Reminder: Object Diagrams and OCL

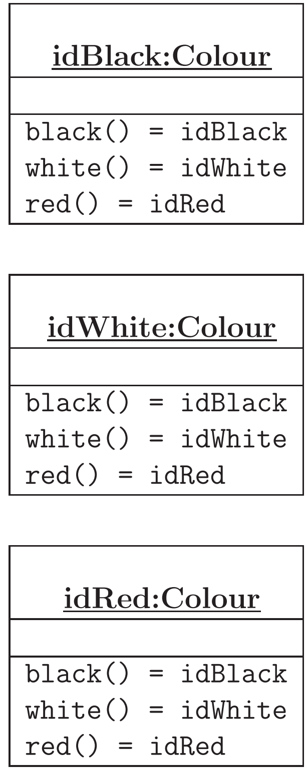


context **Vehicle**
inv: **self.owner.age** $>=$ **18** ✓



Reminder: Object Diagrams and OCL

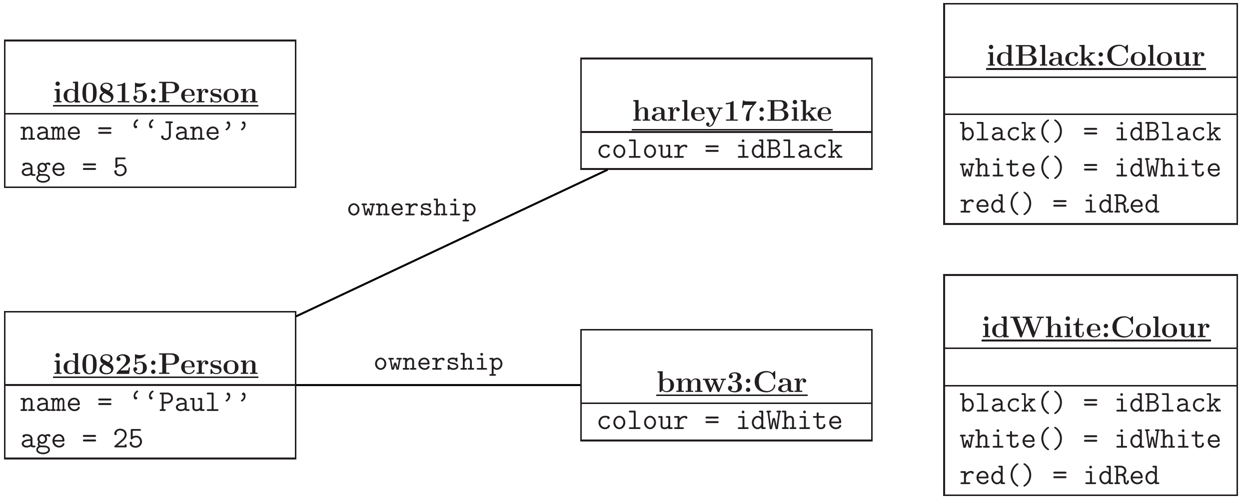


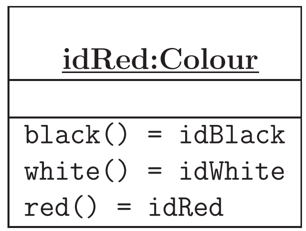


context **Vehicle**
inv: **self.owner.age** $>=$ **18**

$$I(\textbf{Vehicle}) = \{\textbf{harley17}, \textbf{bmw3}\}$$

Reminder: Object Diagrams and OCL





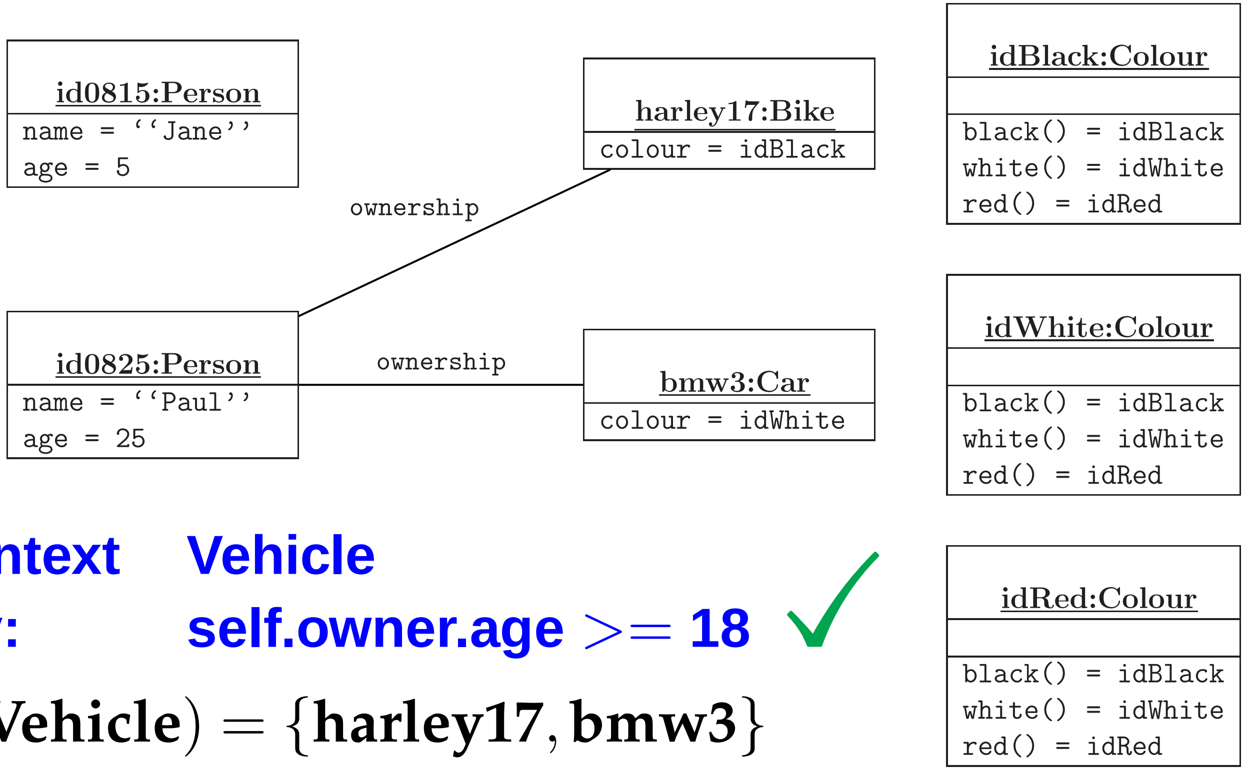
context **Vehicle**

inv: **self.owner.age** $>=$ **18** ✓

$$I(\textbf{Vehicle}) = \{\textbf{harley17}, \textbf{bmw3}\}$$

$$\Rightarrow \textbf{harley17}.I(\textbf{owner}).I(\textbf{age}) \geq 18 \ \textbf{and}\ \textbf{bmw3}.I(\textbf{owner}).I(\textbf{age}) \geq 18$$

Reminder: Object Diagrams and OCL



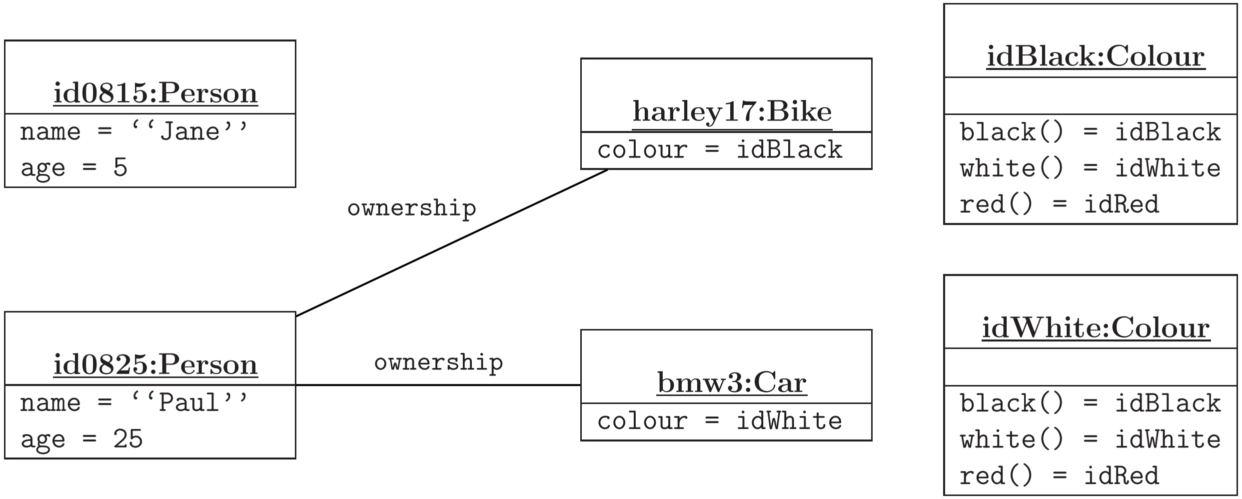
context **Vehicle**
inv: **self.owner.age** $>=$ **18** ✓

$$I(\mathbf{Vehicle}) = \{\mathbf{harley17}, \mathbf{bmw3}\}$$

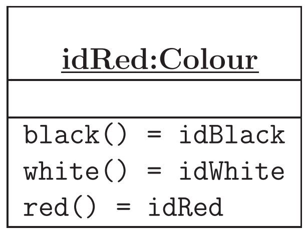
$$\Rightarrow \mathbf{harley17}.I(\mathbf{owner}).I(\mathbf{age}) \geq 18 \text{ and } \mathbf{bmw3}.I(\mathbf{owner}).I(\mathbf{age}) \geq 18$$

$$I(\mathbf{owner}) : \text{Vehicle} \rightarrow \text{Person}$$

Reminder: Object Diagrams and OCL



context **Vehicle**
inv: **self.owner.age** $>=$ **18** ✓

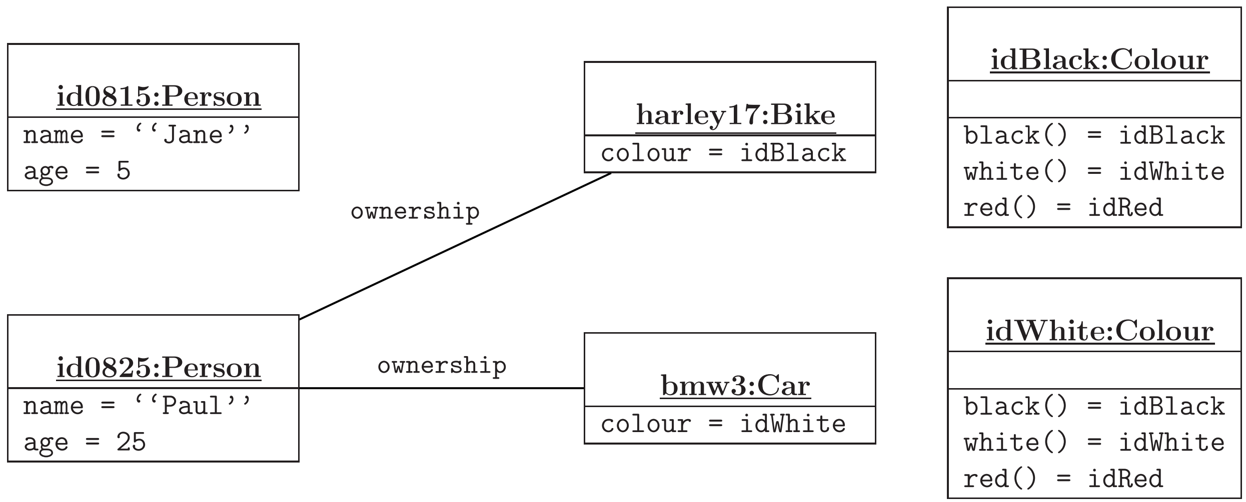


$$I(\textbf{Vehicle}) = \{\textbf{harley17}, \textbf{bmw3}\}$$

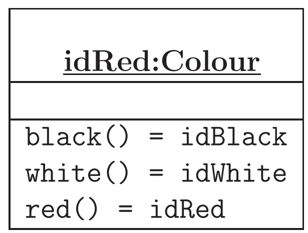
$$\Rightarrow \textbf{harley17}.I(\textbf{owner}).I(\textbf{age}) \geq 18 \text{ and } \textbf{bmw3}.I(\textbf{owner}).I(\textbf{age}) \geq 18$$

$$I(\textbf{owner})(\textbf{harley17}) = I(\textbf{owner})(\textbf{bmw3}) = \textbf{\textit{id0825}}$$

Reminder: Object Diagrams and OCL



context **Vehicle**
inv: **self.owner.age** $>=$ **18** ✓



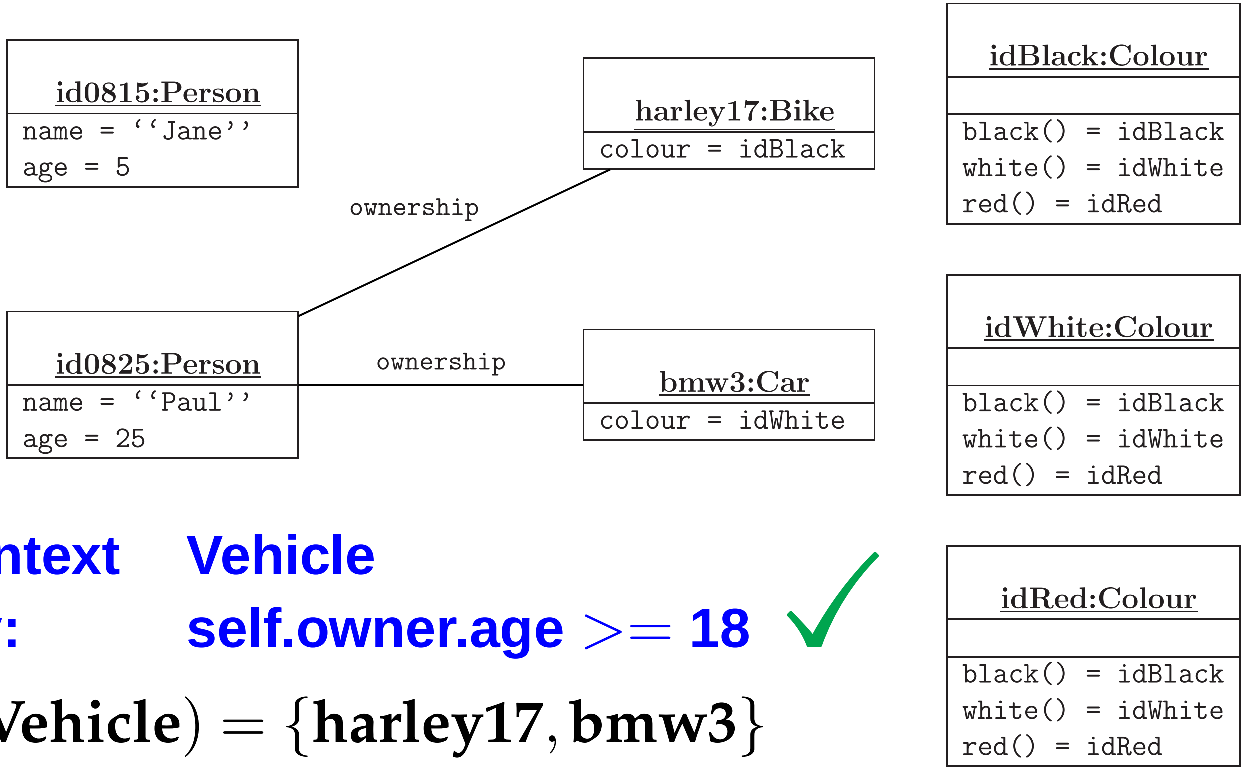
$$I(\textbf{Vehicle}) = \{\textbf{harley17}, \textbf{bmw3}\}$$

$$\Rightarrow \textbf{harley17}.I(\textbf{owner}).I(\textbf{age}) \geq 18 \textbf{ and } \textbf{bmw3}.I(\textbf{owner}).I(\textbf{age}) \geq 18$$

$$I(\textbf{owner})(\textbf{harley17}) = I(\textbf{owner})(\textbf{bmw3}) = \boldsymbol{id0825}$$

$$\Rightarrow \boldsymbol{id0825}.I(\textbf{age}) \geq 18$$

Reminder: Object Diagrams and OCL



context **Vehicle**
inv: **self.owner.age $>=$ 18** ✓

$$I(\textbf{Vehicle}) = \{\textbf{harley17}, \textbf{bmw3}\}$$

$$\Rightarrow \textbf{harley17}.I(\textbf{owner}).I(\textbf{age}) \geq 18 \text{ and } \textbf{bmw3}.I(\textbf{owner}).I(\textbf{age}) \geq 18$$

$$I(\textbf{owner})(\textbf{harley17}) = I(\textbf{owner})(\textbf{bmw3}) = \textbf{\textit{id0825}}$$

$$\Rightarrow \textbf{\textit{id0825}}.I(\textbf{age}) \geq 18$$

$$I(\textbf{age})(\textbf{\textit{id0825}}) = 25 \quad \Rightarrow \quad 25 \geq 18$$ ✓

OCL and Formal Proofs

Snapshots provide formal semantics for UML and OCL

$\Rightarrow$ **can formally prove properties of model and implementation**

OCL and Formal Proofs

Snapshots provide formal semantics for UML and OCL

$\Rightarrow$ **can formally prove properties of model and implementation**

Examples:

- **Invariant of class A implies invariant of class B**

  For each snapshot I: if A's invariant holds in I, then so does B's

  Horizontal verification problem (within specification)

- **Implementation of operation m fulfills its contract**

  For each snapshot I: if precondition of m holds in I, then its postcondition holds in snapshot I' produced by execution of m

  Vertical verification problem (implementation against specification)

Snapshots and States: Static View

Snaphots have static and dynamic part

Snapshots and States: Static View

Snaphots have static and dynamic part

Static (object diagram): objects, attribute values, associations

Static part of snapshot similar to execution state of program

Denote such states with s, set of all states S (infinite!)
Think of one single state as object diagram

Proving horizontal verification problem: state inclusion

Snapshots and States: Static View

Snaphots have static and dynamic part

Static (object diagram): objects, attribute values, associations

Static part of snapshot similar to execution state of program

Denote such states with s, set of all states S (infinite!)
Think of one single state as object diagram

Proving horizontal verification problem: state inclusion

Example: Let $\mathbf{inv}_A$ be invariant of class A, $\mathbf{inv}_B$ invariant of class B

$\{s \in S \mid \mathbf{inv}_A \text{ holds in } s\} \subseteq \{s \in S \mid \mathbf{inv}_B \text{ holds in } s\}$

Snapshots and States: Dynamic View

Program state s = static part of snapshot
Set of all states S

Dynamic part of snapshot:

Semantics of operations m: $\rho(m) : S \rightarrow S$

Operation can be seen as state transformer

For each m and $s \in S$ result state $\rho(m)(s)$
ρ is partial function: programs deterministic, may not terminate

Proving vertical verification problem: state reachability

Snapshots and States: Dynamic View

Program state s = static part of snapshot
Set of all states S

Dynamic part of snapshot:

Semantics of operations m: $\rho(m) : S \to S$

Operation can be seen as state transformer

For each m and $s \in S$ result state $\rho(m)(s)$
ρ is partial function: programs deterministic, may not terminate

Proving vertical verification problem: state reachability

Example: Let pre be precondition, post postcondition of m

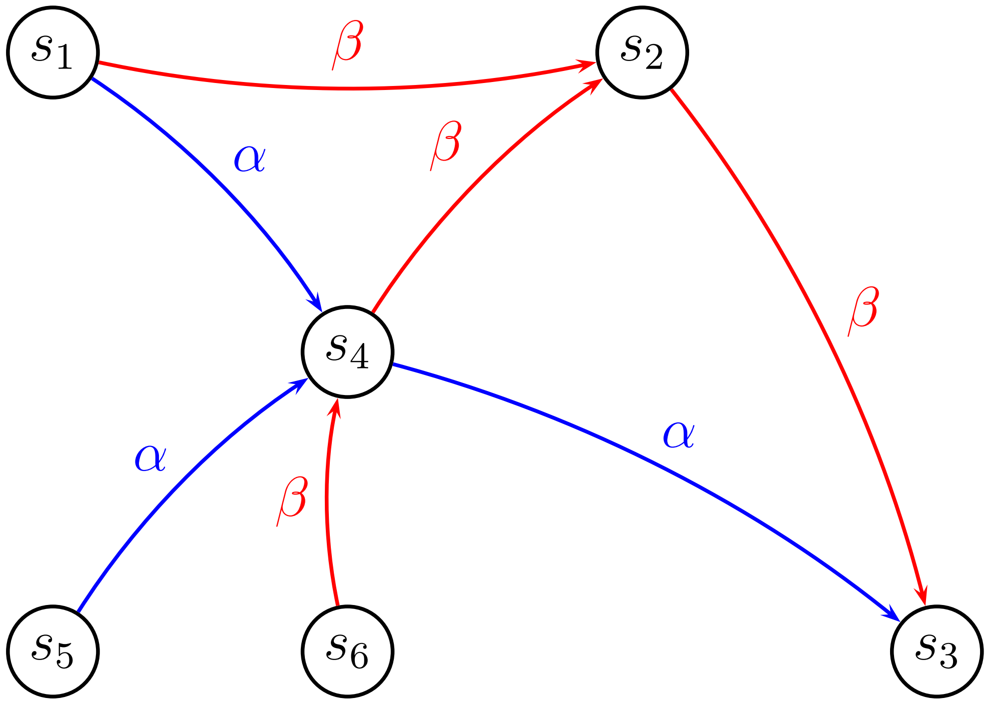
Does post hold in all states $s' \in \{\rho(m)(s) \mid s$ satisfies pre$\}$?

Does post hold in all states s' that can be reached via m from any state s satisfying pre?

Dynamic Part of Snapshots as LTS

(Deterministic) Labelled Transition System (LTS) $K = (S, \rho)$**:**

S **set of states,** ρ : **Method** $\rightarrow (S \rightarrow S)$ **(takes a program and returns a map from** S **to** S**),** $\alpha = \rho(m)$**,** $\beta = \rho(m')$

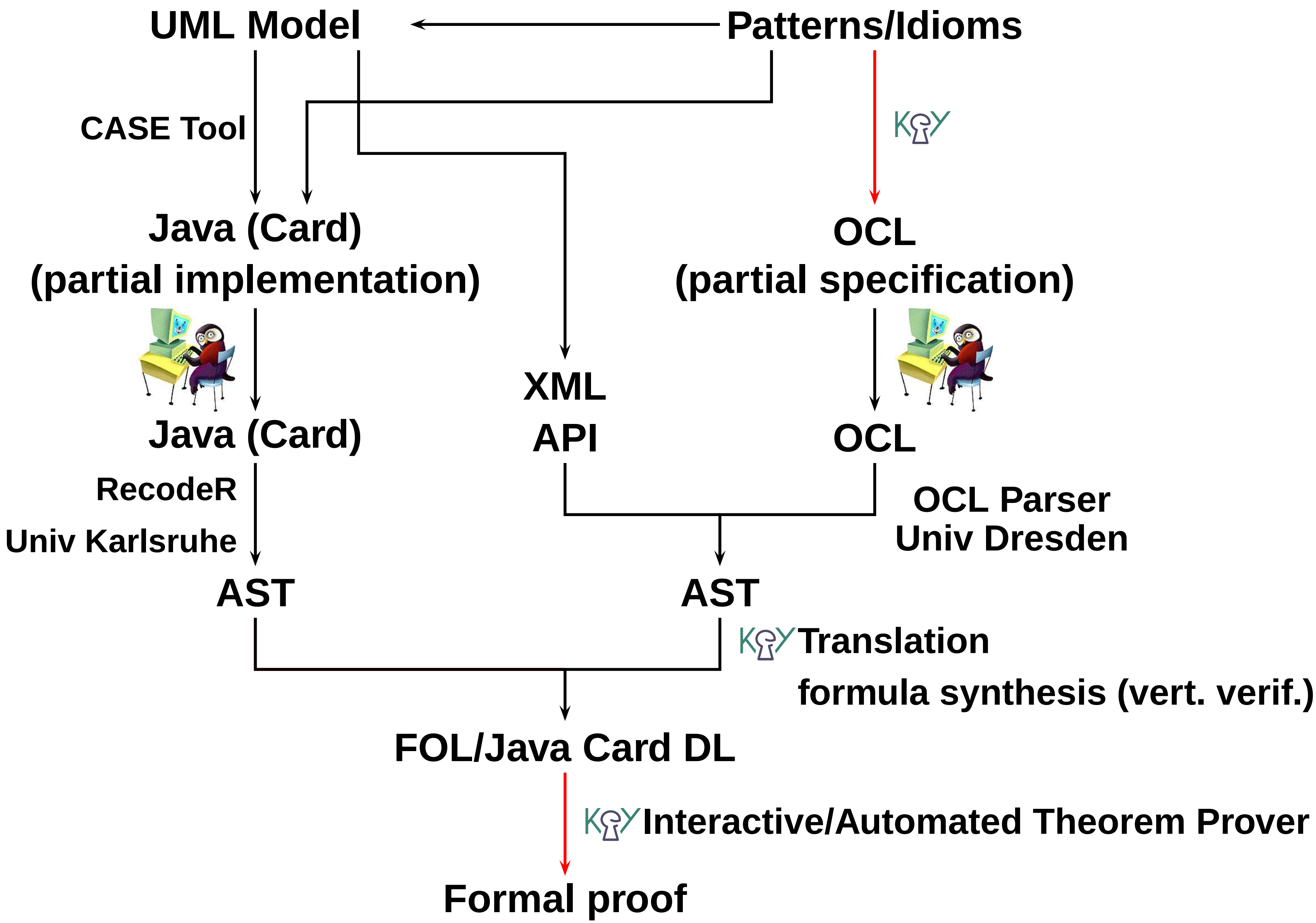


Infinite number of states $\Rightarrow$ **need theorem proving (or approximation)**

Dynamic Part of Snapshots as LTS

- **Each state is a static snapshot (ie, object diagram) with the current objects and values.**
- **If $\rho(m)$ takes, say, state s_1 into s_4, then a directed edge from s_1 to s_4 labelled with $\rho(m)$ is present in K.**
- **$\rho(m)$ is then a (possibly infinite) number of pre-/post execution state pairs.**
- **There is no explicit notion of initial state.**
- **One may consider as initial states those that satisfy the precondition of a distinguished `main` method (and possibly the invariant of its class).**

Encoding Verification Problem in Logic



Why translate OCL into Logic?

Why translate OCL into Logic?

Difficult and expensive to develop theorem prover for a formalism

- **OCL only one of many specification languages (JML, RSL, etc.)**
- **OCL prone to change (1.3, 1.4, 1.5, . . . , 2.0, . . . ?)**
- **First order logic (FOL) well understood, mature tools**
  “FOL in verification like the Reals in Calculus”

Why translate OCL into Logic?

Difficult and expensive to develop theorem prover for a formalism

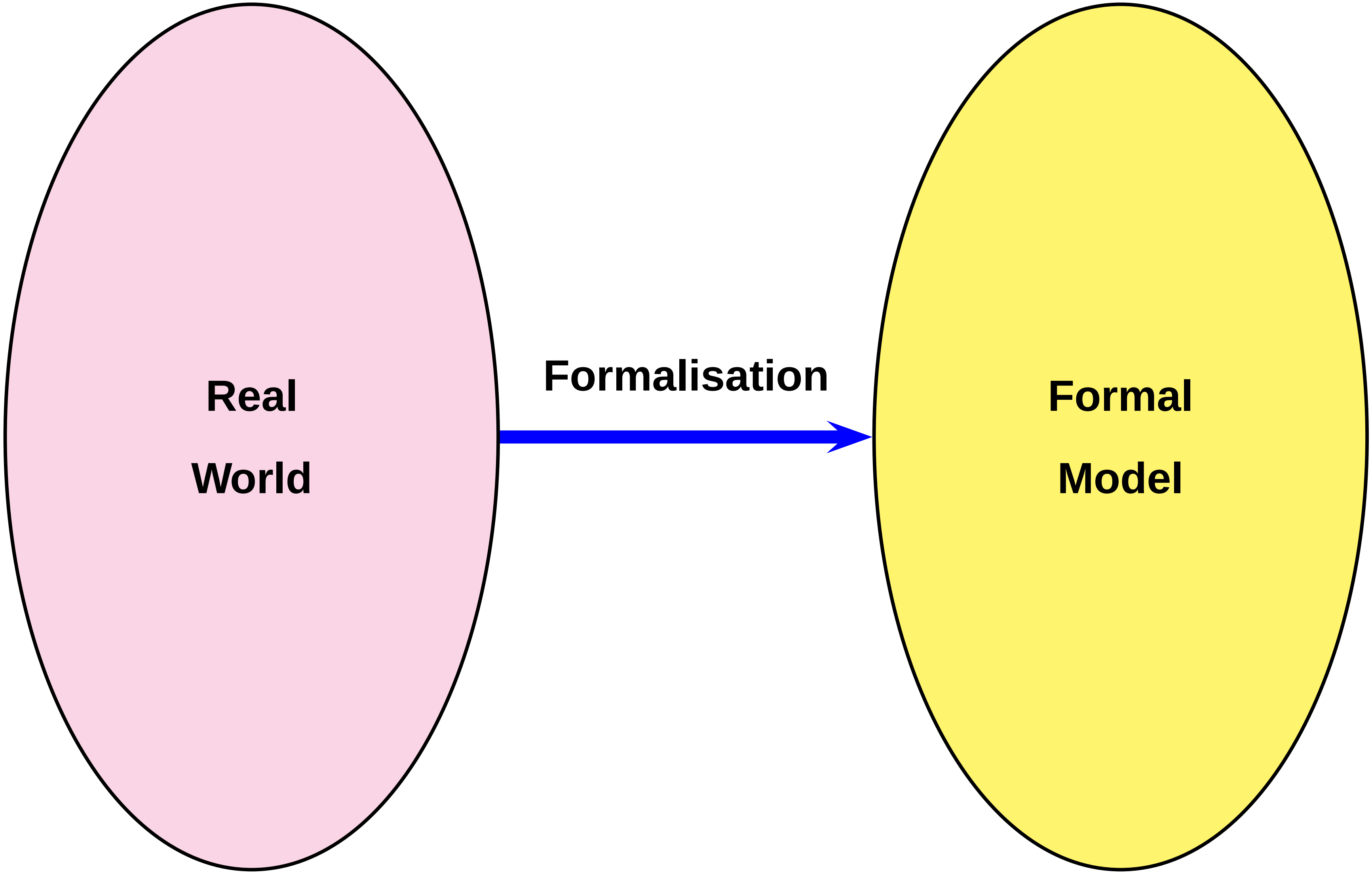
- OCL only one of many specification languages (JML, RSL, etc.)
- OCL prone to change (1.3, 1.4, 1.5, . . . , 2.0, . . . ?)
- First order logic (FOL) well understood, mature tools
  "FOL in verification like the Reals in Calculus"

OCL not designed for verification, programming language independent

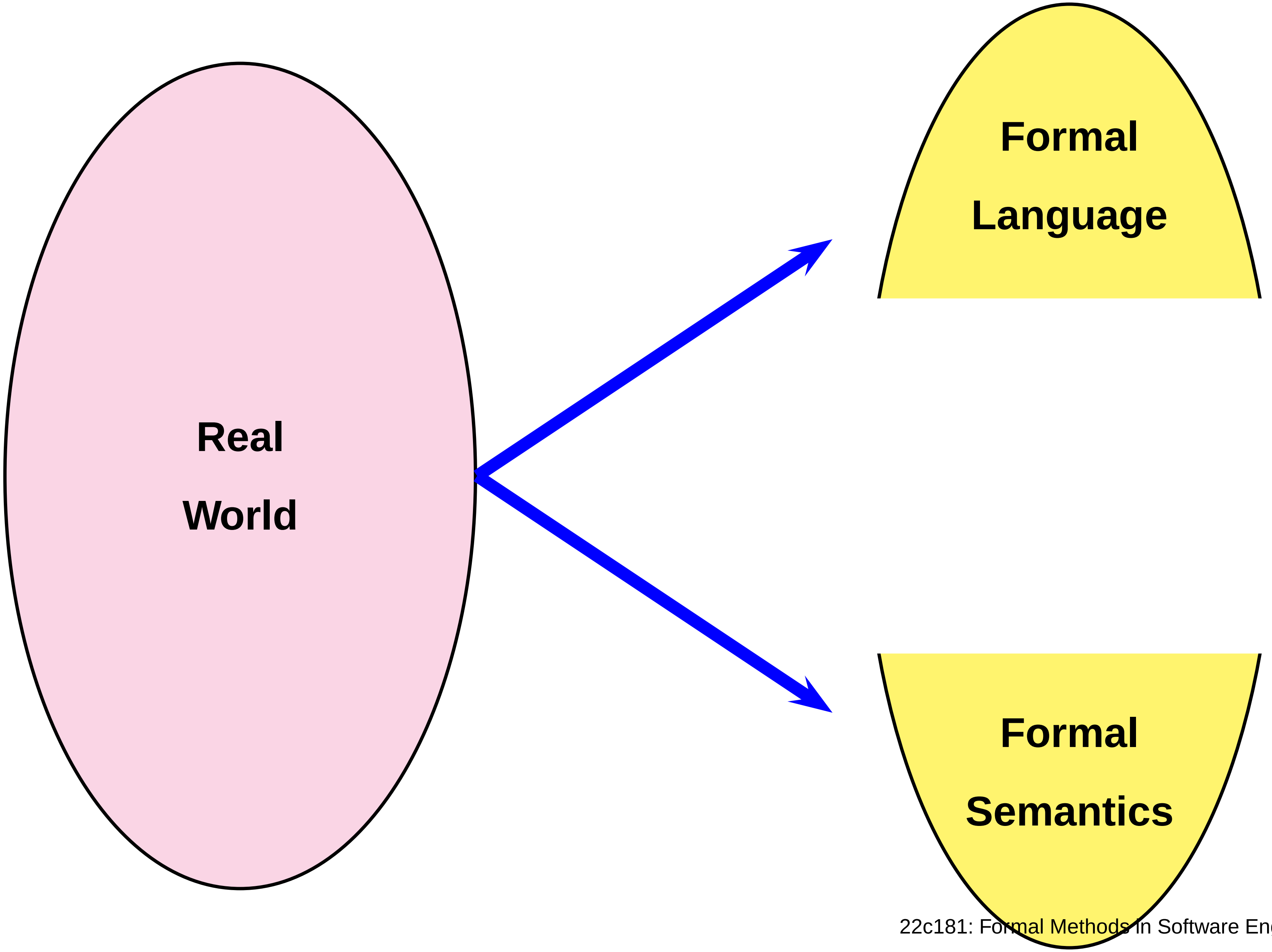
- OCL (UML) doesn't know about implementation of operations
  Need to incorporate Java data types and programs
- OCL not designed to express verification problems
- OCL doesn't know about (class) initialization (<2.0)

Contents

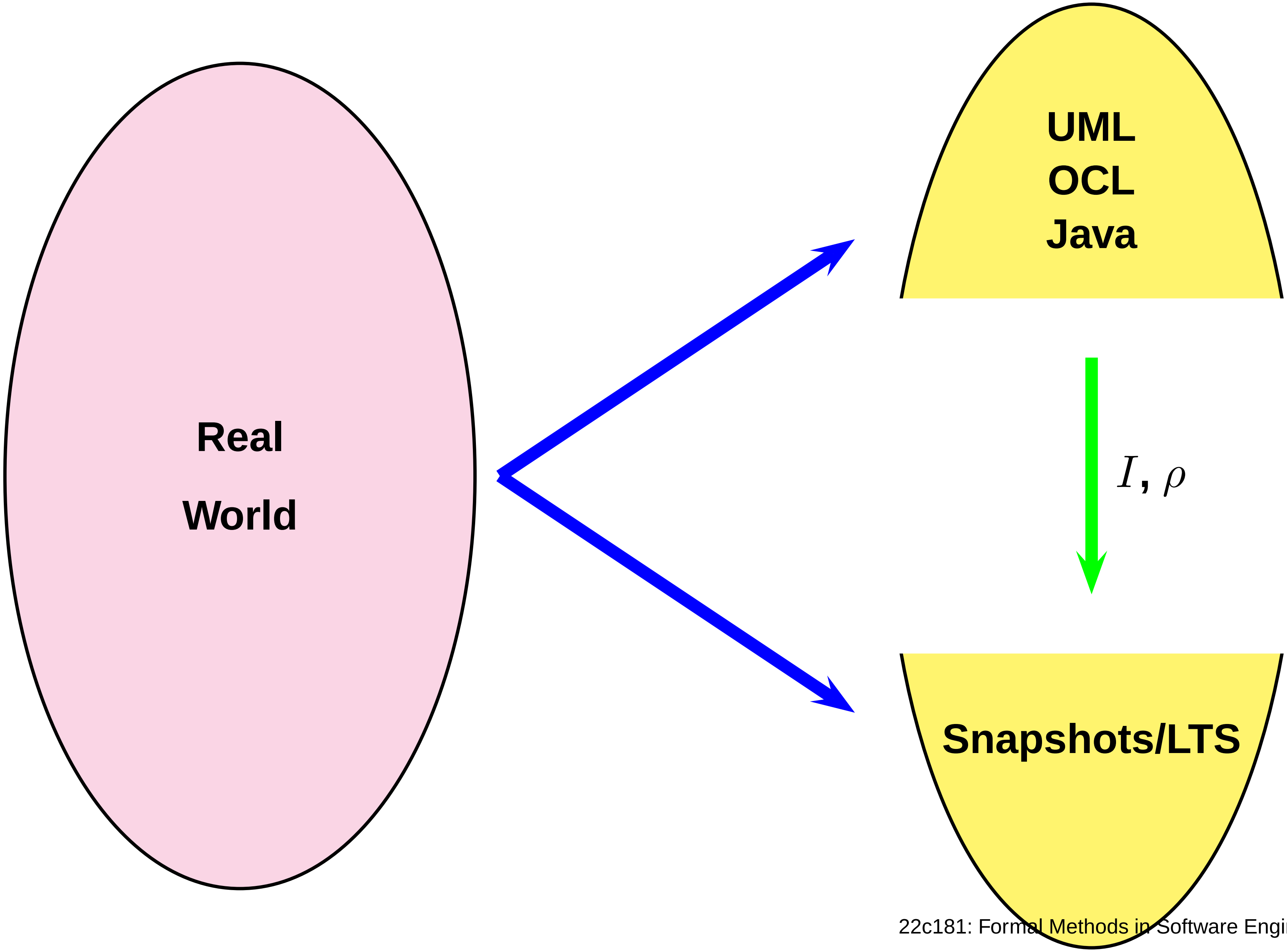
Formalisation



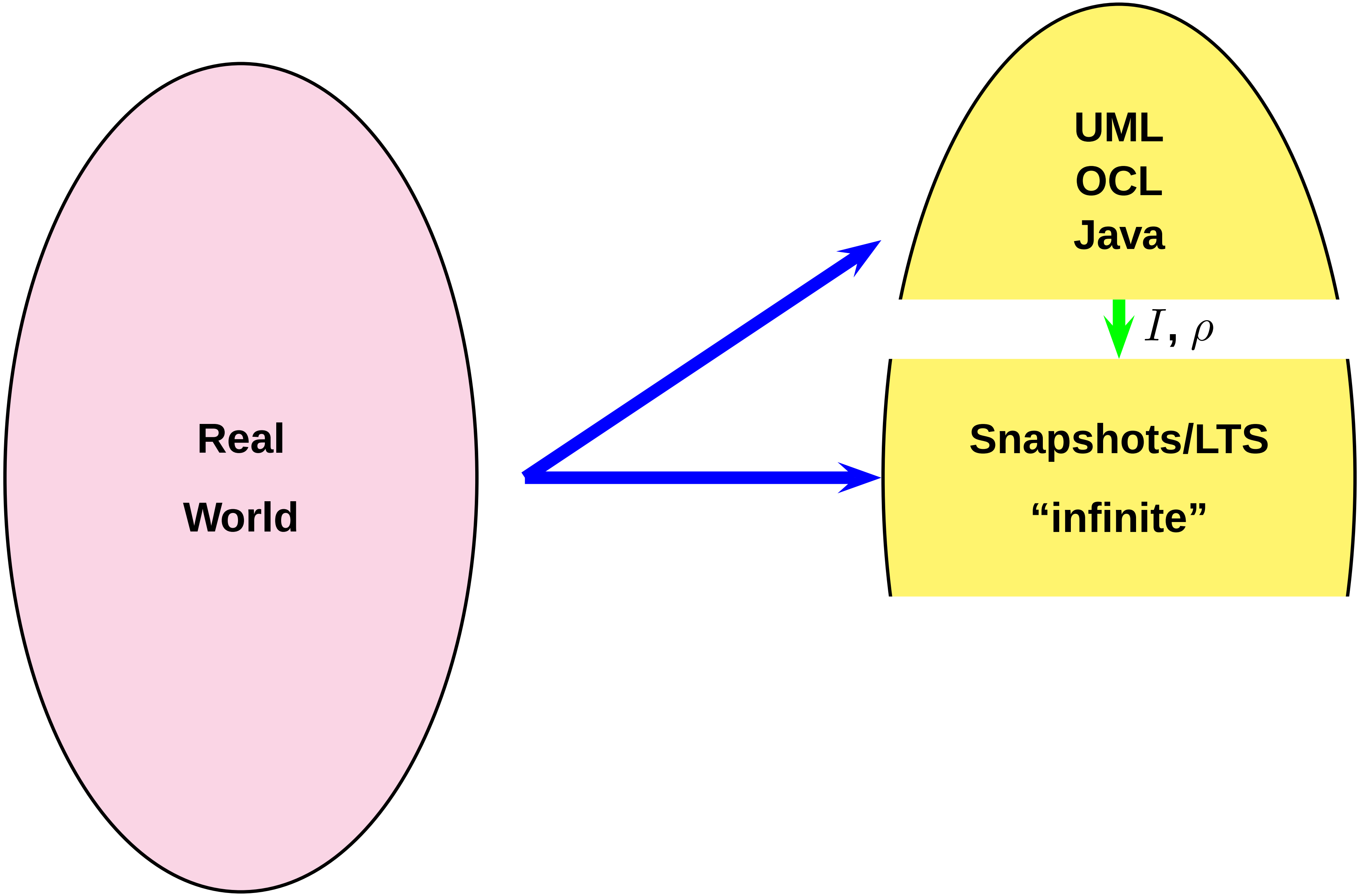
Formalisation



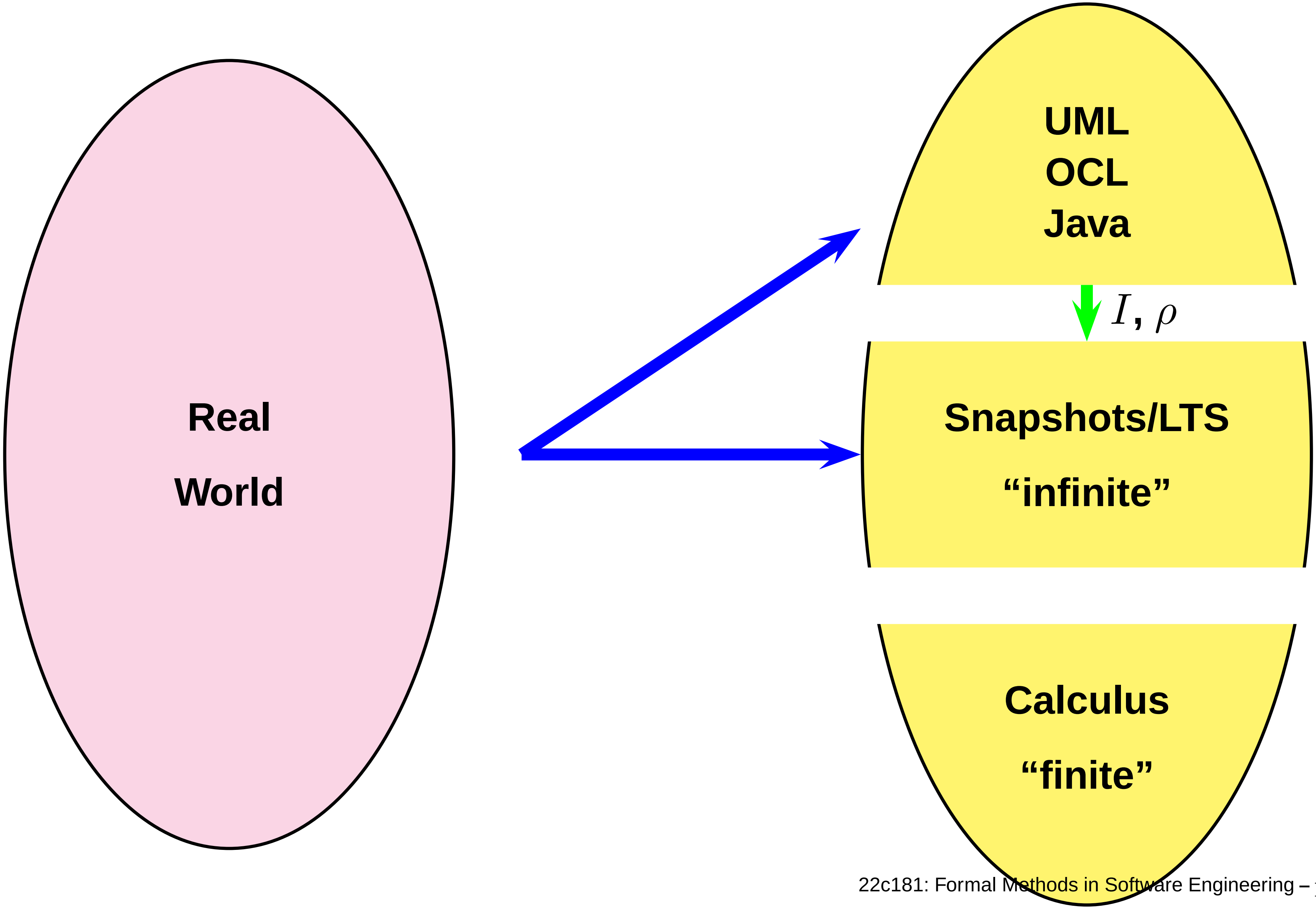
Formalisation



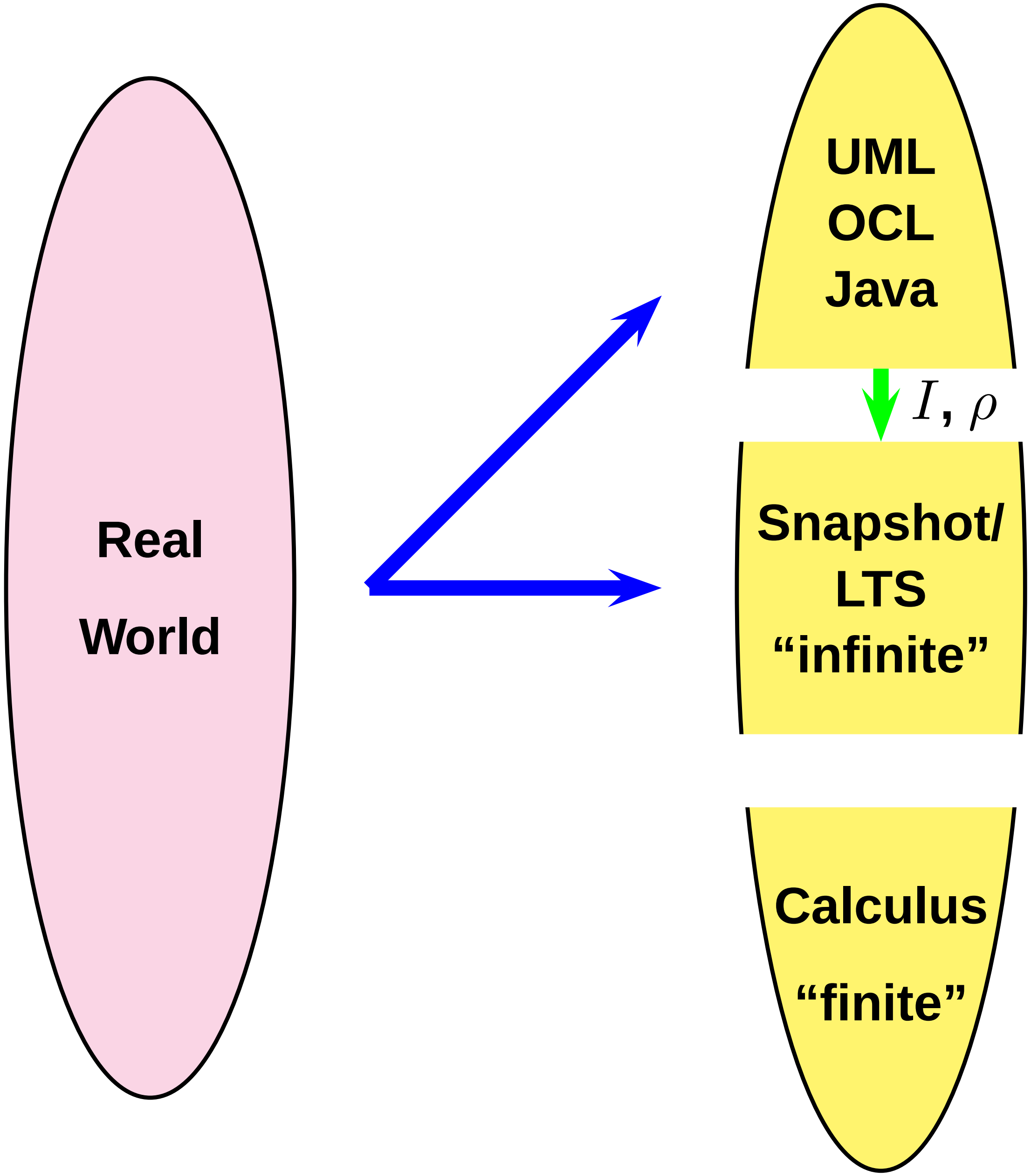
Formalisation



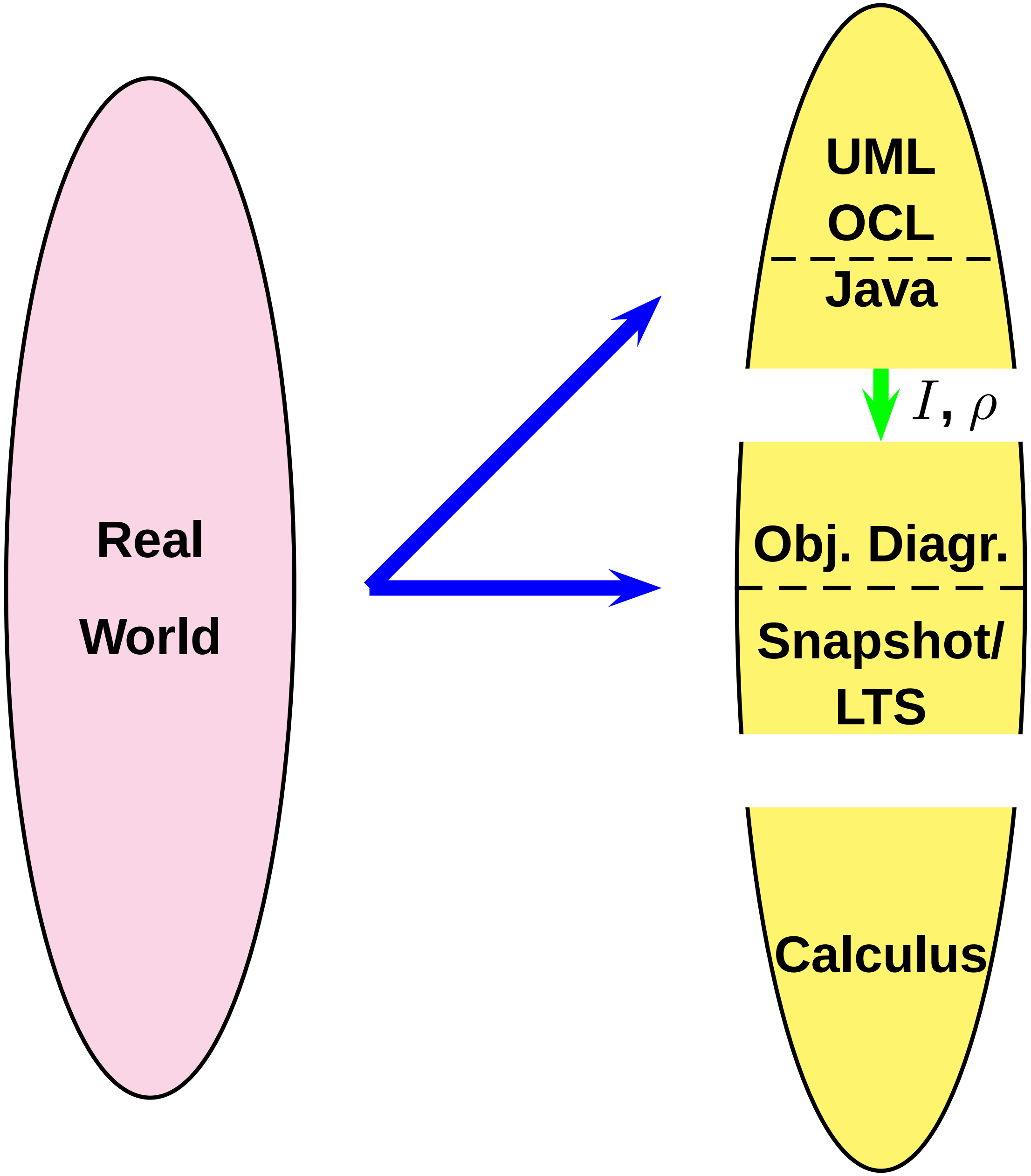
Formalisation



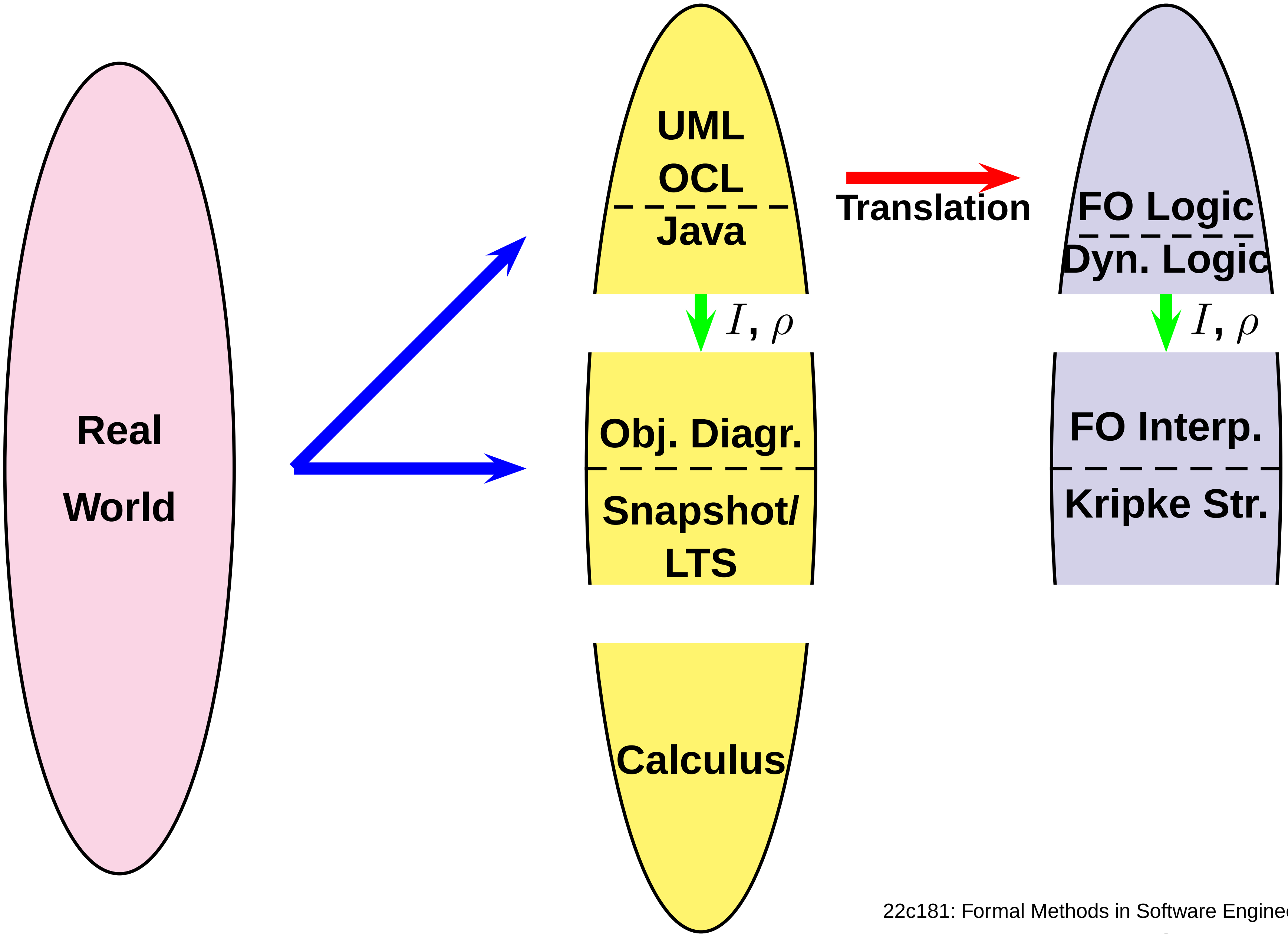
Formal Verification



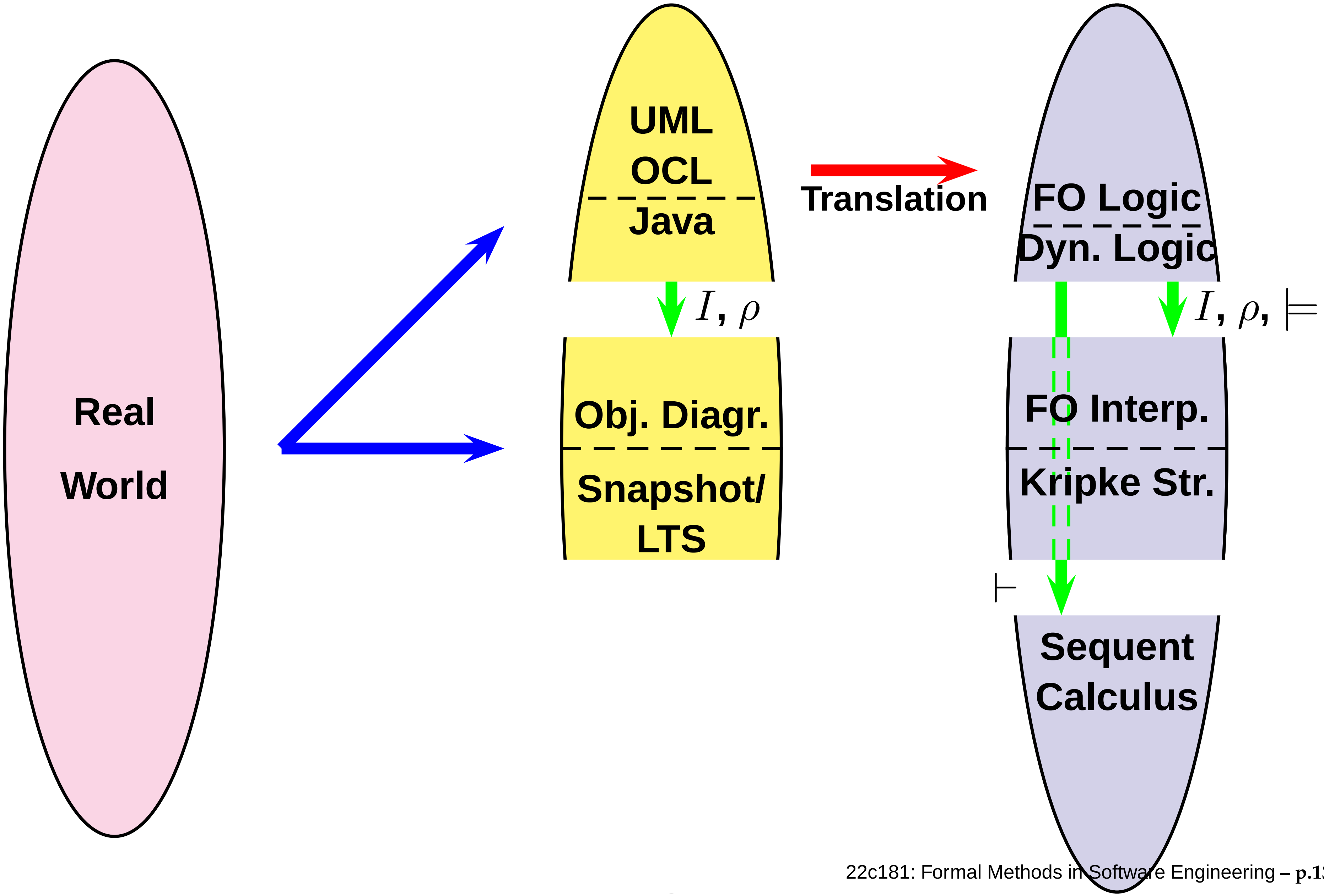
Formal Verification



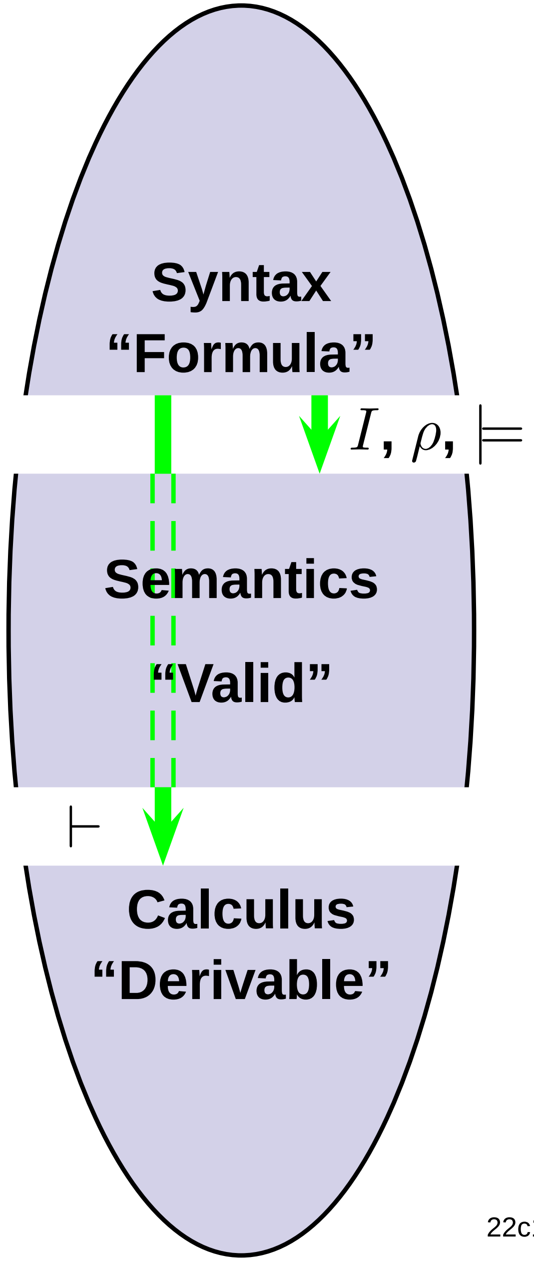
Formal Verification



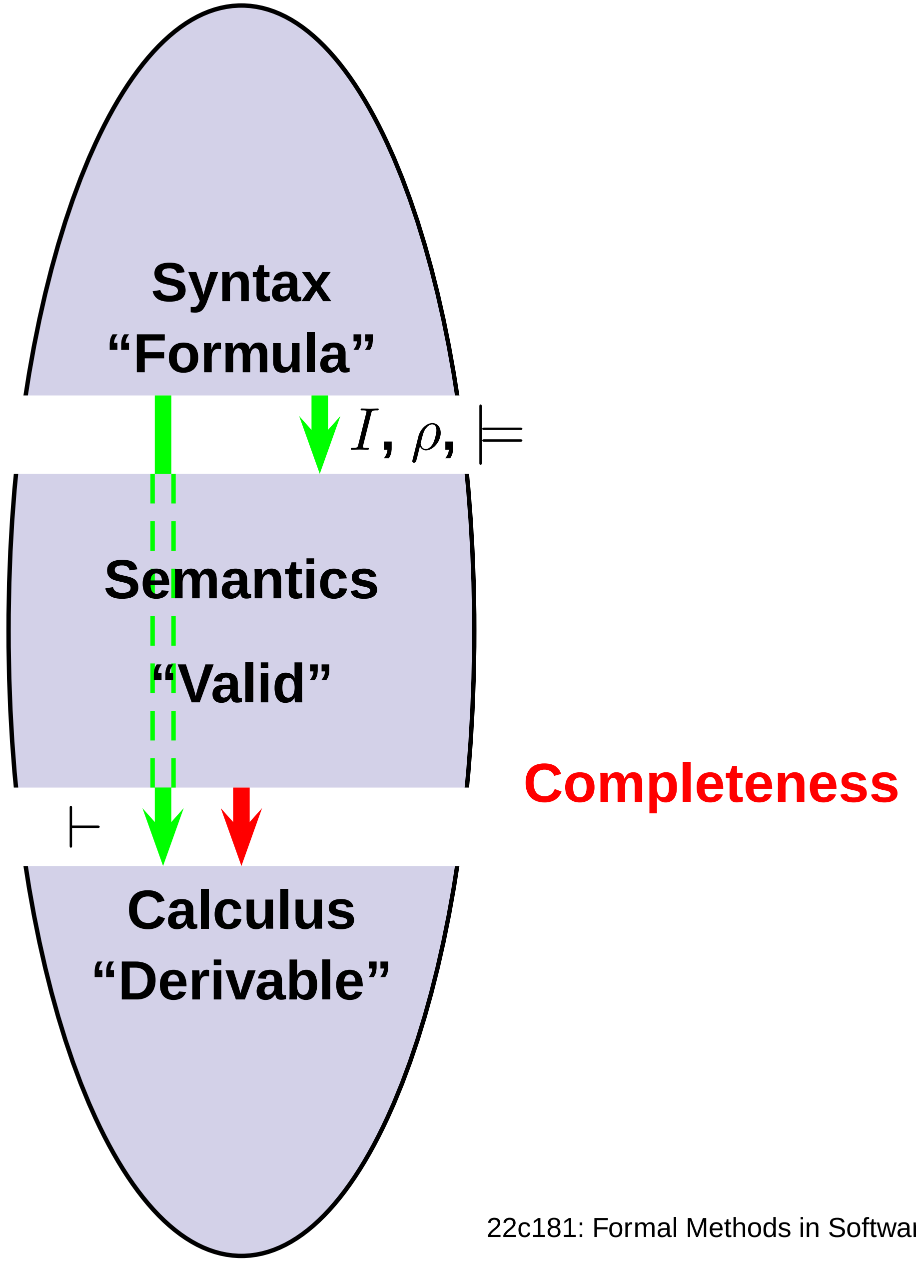
Formal Verification



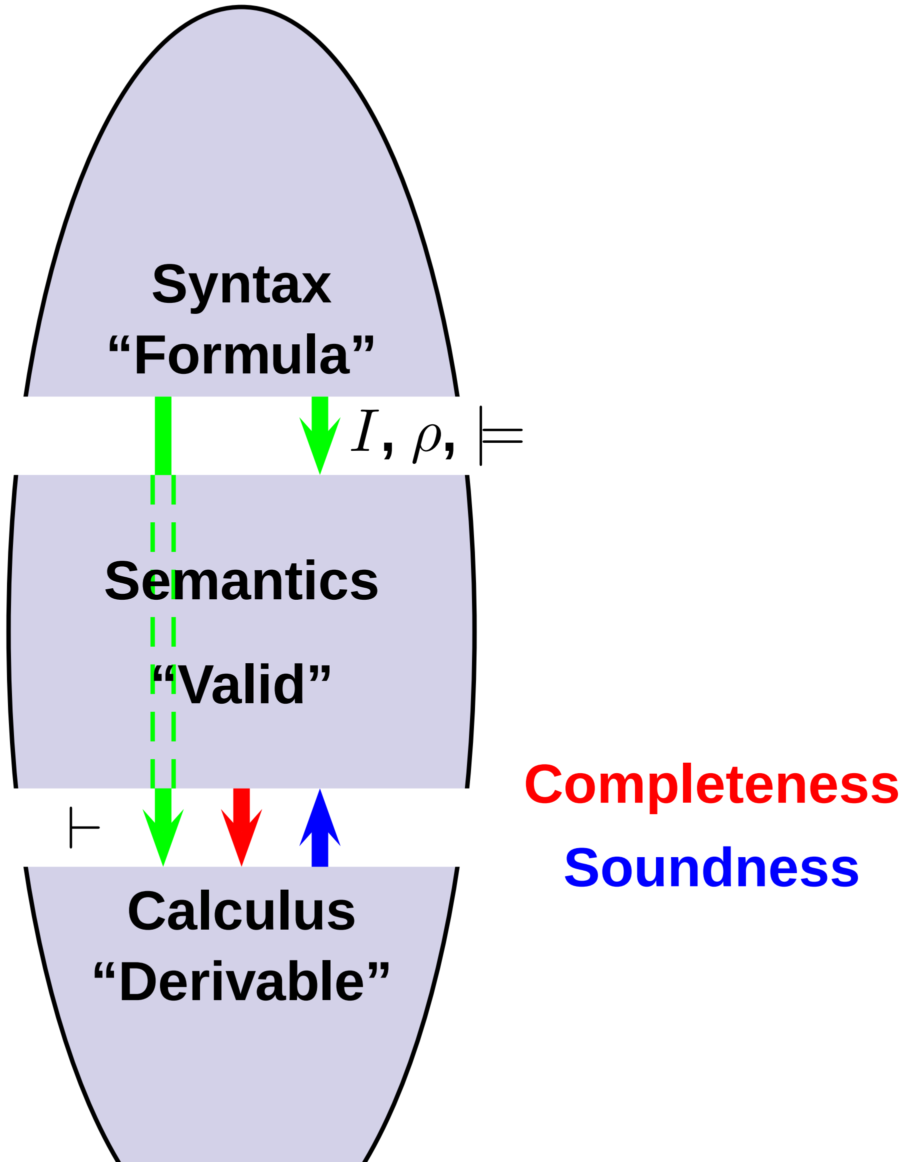
Syntax, Semantics, Calculus



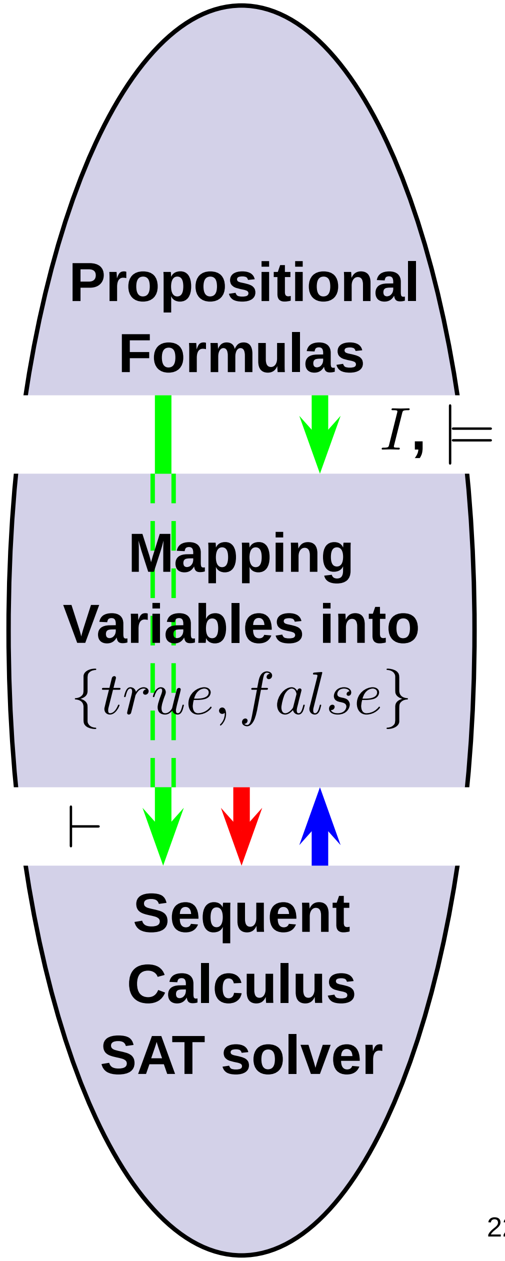
Syntax, Semantics, Calculus



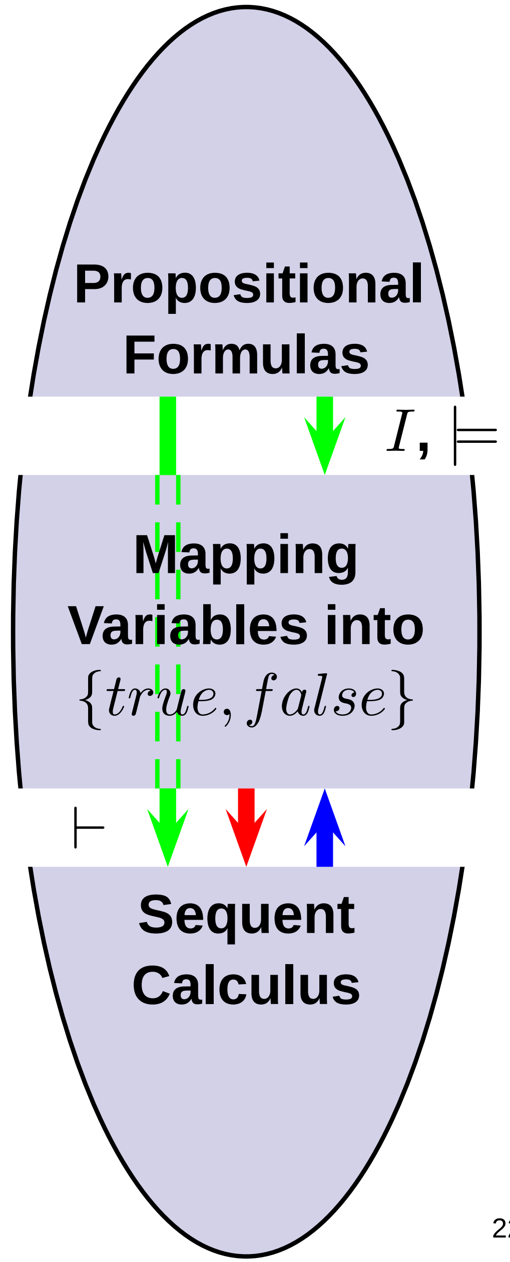
Syntax, Semantics, Calculus



Propositional Logic

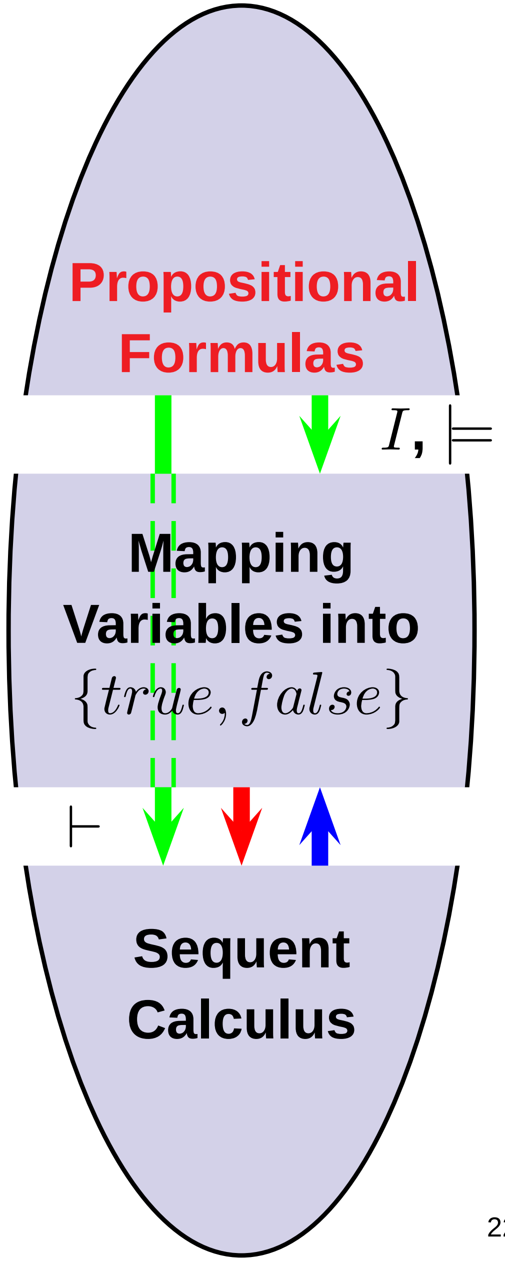


Propositional Logic



Contents

Propositional Logic: Syntax



Syntax of Propositional Logic

- **The Signature:**

  Propositional Variables $\mathcal{P} = \{p_i | i \in I\!N\}$ **with type** $\mathrm{Boolean}$

Syntax of Propositional Logic

- **The Signature:**

  Propositional Variables $\mathcal{P} = \{p_i | i \in I\!N\}$ **with type** $\mathrm{Boolean}$

- **Connectives** $\{$**true**, **false**, **&**, **|**, **!**, **->** , **<->** $\}$

Syntax of Propositional Logic

- **The Signature:**

  Propositional Variables $\mathcal{P} = \{p_i | i \in I\!N\}$ **with type** $\mathrm{Boolean}$

- **Connectives** $\{$**true**, **false**, **&**, **|**, **!**, **->** , **<->** $\}$

Propositional Formulas $\boldsymbol{For}_0$ **(all have type** $\mathrm{Boolean}$**)**

- **Truth constants ‘true’, ‘false’ and variables** $\mathcal{P}$ **are formulas**

Syntax of Propositional Logic

- **The Signature:**

  Propositional Variables $\mathcal{P} = \{p_i | i \in I\!N\}$ **with type** Boolean

- **Connectives** $\{$**true**, **false**, **&**, **|**, **!**, **->** , **<->** $\}$

Propositional Formulas $\boldsymbol{For}_0$ **(all have type** Boolean**)**

- **Truth constants 'true', 'false' and variables** $\mathcal{P}$ **are formulas**

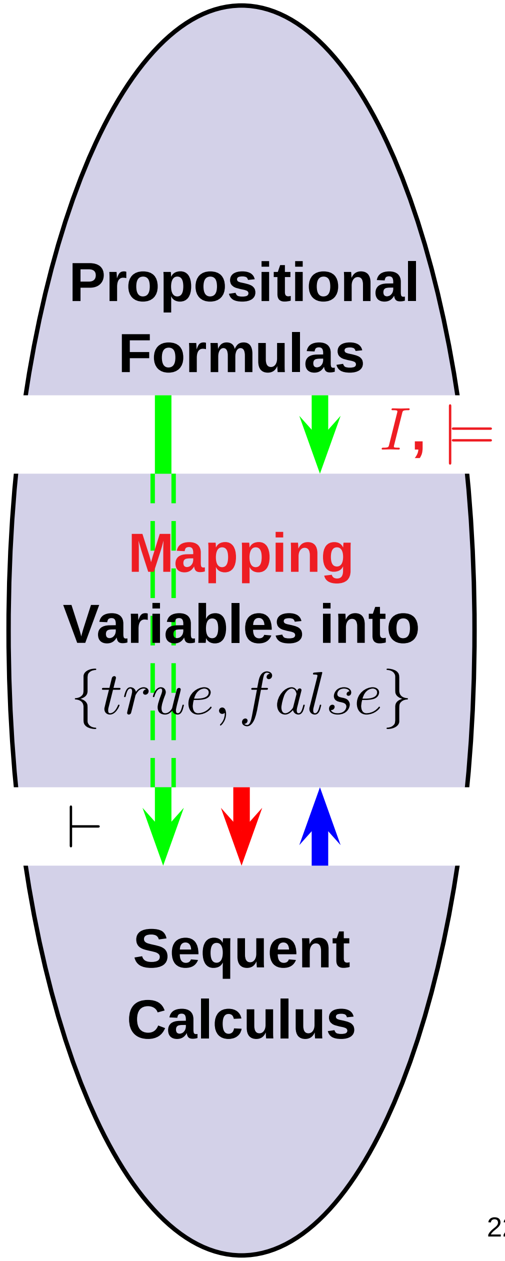
- **If** G **and** H **are formulas then**

  $$!G, \quad (G \,\&\, H), \quad (G \mid H), \quad (G \text{ -> } H), \quad (G \text{ <-> } H)$$

  are also formulas

- **There are no other formulas (inductive definition)**

Propositional Logic: Semantics



Semantics of Propositional Logic

Interpretation $\mathcal{I}$

Assigns a truth value to each propositional variable

$$\mathcal{I} : \mathcal{P} \rightarrow \{true, false\}$$

Semantics of Propositional Logic

Interpretation $\mathcal{I}$

Assigns a truth value to each propositional variable

$$\mathcal{I} : \mathcal{P} \rightarrow \{true, false\}$$

Valuation function $val_{\mathcal{I}}$**: extension of** $\mathcal{I}$ **to** $\boldsymbol{For}_0$

$$val_{\mathcal{I}} : \boldsymbol{For}_0 \ \rightarrow \ \{true, false\}$$

Semantics of Propositional Logic

Interpretation $\mathcal{I}$

Assigns a truth value to each propositional variable

$$\mathcal{I} : \mathcal{P} \rightarrow \{true, false\}$$

Valuation function $val_{\mathcal{I}}$**: extension of** $\mathcal{I}$ **to** $\boldsymbol{For}_0$

$$val_{\mathcal{I}} : \boldsymbol{For}_0 \rightarrow \{true, false\}$$

$$val_{\mathcal{I}}(p_i) = \mathcal{I}(p_i)$$

$$val_{\mathcal{I}}(\textbf{true}) = true$$

$$val_{\mathcal{I}}(\textbf{false}) = false$$

$$val_{\mathcal{I}}(G \textbf{ -> } H) = \begin{cases} true & \textbf{if } val_{\mathcal{I}}(G) = false \textbf{ or} \\ & \quad val_{\mathcal{I}}(H) = true \\ false & otherwise \end{cases}$$

etc.

$\mathcal{I}$ **satisfies** G **if** $val_{\mathcal{I}}(G) = true$**; otherwise, it falsifies** G**.**

Example

Formula

$$p \text{ -> } (q \text{ -> } p)$$

Example

Formula

$$p \textbf{ -> } (q \textbf{ -> } p)$$

Interpretation (one of four that are possible)

$\mathcal{I}(p) = true$
$\mathcal{I}(q) = false$

Example

Formula

$$p \textbf{ -> } (q \textbf{ -> } p)$$

Interpretation (one of four that are possible)

$\mathcal{I}(p) = true$
$\mathcal{I}(q) = false$

Valuation

$$val_{\mathcal{I}}(q \textbf{ -> } p) \quad = \quad true$$

Example

Formula

$$p \textbf{ -> } (q \textbf{ -> } p)$$

Interpretation (one of four that are possible)

$\mathcal{I}(p) = true$
$\mathcal{I}(q) = false$

Valuation

$$val_{\mathcal{I}}(\ q \textbf{ -> } p\ ) \quad = \quad true$$

$$val_{\mathcal{I}}(\ p \textbf{ -> } (q \textbf{ -> } p)\ ) \quad = \quad true$$

Semantic Notions

Let $G \in \boldsymbol{For}_0$, $\Gamma \subset \boldsymbol{For}_0$

- **Validity Relation $\models$**

  G is valid in $\mathcal{I}$ iff $val_{\mathcal{I}}(G) = true$ (write: $\mathcal{I} \models G$)

  A formula that is valid in some interpretation is satisfiable

- **Γ entails G ($\Gamma \models G$) iff for all interpretations $\mathcal{I}$:**

  $$\text{If } \mathcal{I} \models H \textbf{ for all } H \in \Gamma \textbf{ then also } \mathcal{I} \models G$$

- **If G is valid in any interpretation, i.e**

  $$\emptyset \models G \quad (\text{short}: \models G)$$

  then G is called logically valid

Propositional Logic Examples

$$p \mathbin{\&} ((!p) \mid q)$$

Satisfiable?

Propositional Logic Examples

$$p \mathbin{\textbf{\&}} ((\textbf{!}p) \mathbin{\textbf{|}} q)$$

Satisfiable? **Yes**

Propositional Logic Examples

$$p \;\textbf{\&}\; ((\textbf{!}p) \;\textbf{|}\; q)$$

Satisfiable? **Yes**

Satisfying Interpretation?

Propositional Logic Examples

$$p \,\textbf{\&}\, ((\textbf{!}p) \,\textbf{|}\, q)$$

Satisfiable? **Yes**

Satisfying Interpretation? $\mathcal{I}(p) = true, \mathcal{I}(q) = true$

Propositional Logic Examples

$$p \,\&\, ((!p) \mid q)$$

Satisfiable? **Yes**

Satisfying Interpretation? $\mathcal{I}(p) = true, \mathcal{I}(q) = true$

$$p \,\&\, ((!p) \mid q) \models q \mid r$$

Does this hold?

Propositional Logic Examples

$$p \mathbin{\&} ((!p) \mid q)$$

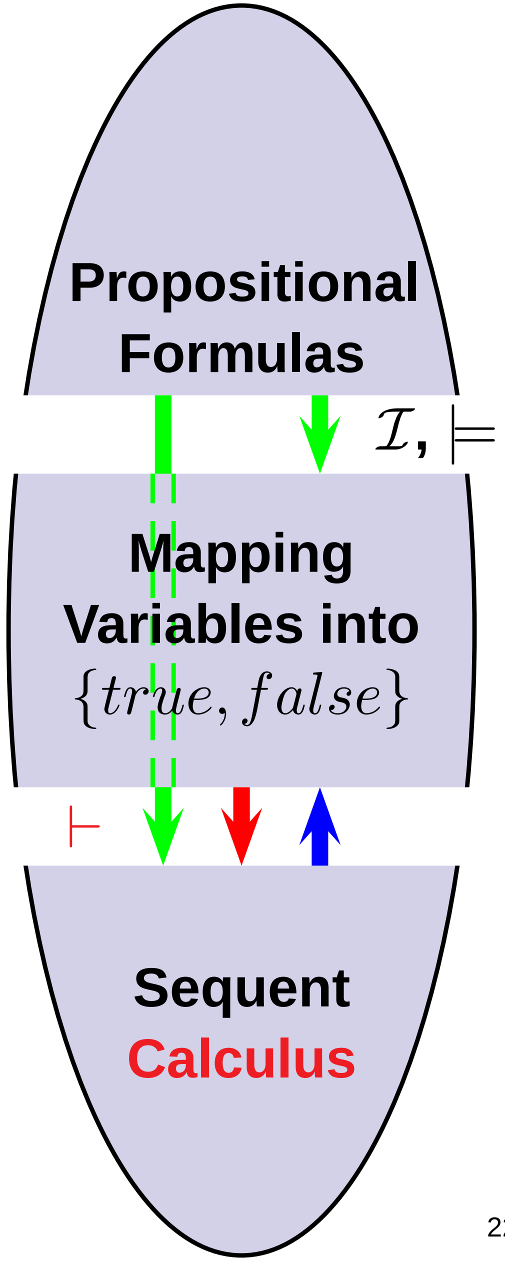
Satisfiable? **Yes**

Satisfying Interpretation? $\mathcal{I}(p) = true, \mathcal{I}(q) = true$

$$p \mathbin{\&} ((!p) \mid q) \models q \mid r$$

Does this hold? **Yes.** **Why?**

Propositional Logic



Reasoning by Syntactic Transformation

Establish $\models G$ by finite syntactic transformations of G

Reasoning by Syntactic Transformation

Establish $\models G$ by finite syntactic transformations of G

(Logic) Calculus: a set of syntactic transformation rules $\mathcal{R}$ defining a property $\vdash$ over For_0 such that $\models G$ iff $\vdash G$ (G is derivable)

$\models G$ implies $\vdash G$ (Completeness) **$\vdash G$ implies $\models G$ (Soundness)**

Reasoning by Syntactic Transformation

Establish $\models G$ by finite syntactic transformations of G

(Logic) Calculus: a set of syntactic transformation rules $\mathcal{R}$ defining a property $\vdash$ over For_0 such that $\models G$ iff $\vdash G$ (G is derivable)

$\models G$ implies $\vdash G$ (Completeness) **$\vdash G$ implies $\models G$ (Soundness)**

Sequent Calculus based on notion of sequent

$$\underbrace{\psi_1, \ldots, \psi_m}_{Antecedent} \quad \texttt{==>} \quad \underbrace{\phi_1, \ldots, \phi_n}_{Succedent}$$

has same semantics as

$$(\psi_1 \,\&\, \cdots \,\&\, \psi_m) \quad \texttt{->} \quad (\phi_1 \mid \cdots \mid \phi_n)$$

$$\{\psi_1, \ldots, \psi_m\} \quad \models \quad \phi_1 \mid \cdots \mid \phi_n$$

Notation for Sequents

$$\psi_1, \ldots, \psi_m \quad \textbf{==>} \quad \phi_1, \ldots, \phi_n$$

Consider antecedent/succedent as sets of formulas, may be empty

Use schema variables $\Gamma, \phi, \ldots$ that match (sets of) formulas
Characterize infinitely many formulas with a single sequent

$$\Gamma \quad \textbf{==>} \quad \Delta, \phi \,\&\, \psi$$

Matches any sequent with occurrence of conjunction in succedent

Call $\phi \,\&\, \psi$ main formula and Γ, Δ side formulas of sequent

Any sequent of the form Γ, ϕ ==> Δ, ϕ is logically valid, and is called an axiom

Sequent Calculus Rules

Basic idea: write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{RULE NAME}\ \frac{\overbrace{\Gamma_1 ==> \Delta_1 \quad \cdots \quad \Gamma_r ==> \Delta_r}^{\text{Premisses}}}{\underbrace{\Gamma ==> \Delta}_{\text{Conclusion}}}$$

Sequent Calculus Rules

Basic idea: **write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible**

$$\text{RULE NAME}\ \frac{\overbrace{\Gamma_1 ==> \Delta_1 \quad \cdots \quad \Gamma_r ==> \Delta_r}^{\text{Premisses}}}{\underbrace{\Gamma ==> \Delta}_{\text{Conclusion}}}$$

Example

$$\textbf{AND\_RIGHT}\ \frac{\Gamma\ \textbf{==>}\ \phi, \Delta \quad \Gamma\ \textbf{==>}\ \psi, \Delta}{\Gamma\ \textbf{==>}\ \phi\ \textbf{\&}\ \psi, \Delta}$$

Sequent Calculus Rules

Basic idea: **write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible**

$$\text{RULE NAME}\ \frac{\overbrace{\Gamma_1 ==> \Delta_1 \quad \cdots \quad \Gamma_r ==> \Delta_r}^{\text{Premisses}}}{\underbrace{\Gamma ==> \Delta}_{\text{Conclusion}}}$$

Example

$$\text{AND\_RIGHT}\ \frac{\Gamma \ \texttt{==>}\ \phi, \Delta \quad \Gamma \ \texttt{==>}\ \psi, \Delta}{\Gamma \ \texttt{==>}\ \phi \,\&\, \psi, \Delta}$$

Rules can have zero premisses (iff conclusion is valid, eg. an axiom)

Sequent Calculus Rules

Basic idea: **write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible**

$$\text{RULE NAME}\;\frac{\overbrace{\Gamma_1 \;==>\; \Delta_1 \quad \cdots \quad \Gamma_r \;==>\; \Delta_r}^{\text{Premisses}}}{\underbrace{\Gamma \;==>\; \Delta}_{\text{Conclusion}}}$$

Example

$$\textbf{AND\_RIGHT}\;\frac{\Gamma \;\textbf{==>}\; \phi, \Delta \quad \Gamma \;\textbf{==>}\; \psi, \Delta}{\Gamma \;\textbf{==>}\; \phi \,\&\, \psi, \Delta}$$

- **A rule is sound if every interpretation that satisfies each premiss of the rule also satisfies its conclusion (essential property)**

Sequent Calculus Rules

Basic idea: **write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible**

$$\text{RULE NAME}\ \frac{\overbrace{\Gamma_1 \ \text{==>}\ \Delta_1 \quad \cdots \quad \Gamma_r \ \text{==>}\ \Delta_r}^{\text{Premisses}}}{\underbrace{\Gamma \ \text{==>}\ \Delta}_{\text{Conclusion}}}$$

Example

$$\text{AND\_RIGHT}\ \frac{\Gamma \ \text{==>}\ \phi, \Delta \quad \Gamma \ \text{==>}\ \psi, \Delta}{\Gamma \ \text{==>}\ \phi \,\&\, \psi, \Delta}$$

- **A rule is sound if every interpretation that satisfies each premiss of the rule also satisfies its conclusion (essential property)**
- **A rule is complete if every interpretation that satisfies its conclusion also satisfies each of its premisses (desirable property)**

Rules of Propositional Sequent Calculus

main	left side (work on antecedent)	right side (work on succedent)
not	$$\frac{\Gamma \;\; \textbf{==>} \;\; \phi, \Delta}{\Gamma, \textbf{!}\phi \;\; \textbf{==>} \;\; \Delta}$$	$$\frac{\Gamma, \phi \;\; \textbf{==>} \;\; \Delta}{\Gamma \;\; \textbf{==>} \;\; \textbf{!}\phi, \Delta}$$

Rules of Propositional Sequent Calculus

main	left side (work on antecedent)	right side (work on succedent)
not	$\dfrac{\Gamma \text{ ==> } \phi, \Delta}{\Gamma, !\phi \text{ ==> } \Delta}$	$\dfrac{\Gamma, \phi \text{ ==> } \Delta}{\Gamma \text{ ==> } !\phi, \Delta}$
and	$\dfrac{\Gamma, \phi, \psi \text{ ==> } \Delta}{\Gamma, \phi \,\&\, \psi \text{ ==> } \Delta}$	$\dfrac{\Gamma \text{ ==> } \phi, \Delta \quad \Gamma \text{ ==> } \psi, \Delta}{\Gamma \text{ ==> } \phi \,\&\, \psi, \Delta}$

Rules of Propositional Sequent Calculus

main	left side (work on antecedent)	right side (work on succedent)
not	$\dfrac{\Gamma \;\texttt{==>}\; \phi, \Delta}{\Gamma, !\phi \;\texttt{==>}\; \Delta}$	$\dfrac{\Gamma, \phi \;\texttt{==>}\; \Delta}{\Gamma \;\texttt{==>}\; !\phi, \Delta}$
and	$\dfrac{\Gamma, \phi, \psi \;\texttt{==>}\; \Delta}{\Gamma, \phi \,\&\, \psi \;\texttt{==>}\; \Delta}$	$\dfrac{\Gamma \;\texttt{==>}\; \phi, \Delta \quad \Gamma \;\texttt{==>}\; \psi, \Delta}{\Gamma \;\texttt{==>}\; \phi \,\&\, \psi, \Delta}$
or	$\dfrac{\Gamma, \phi \;\texttt{==>}\; \Delta \quad \Gamma, \psi \;\texttt{==>}\; \Delta}{\Gamma, \phi \mid \psi \;\texttt{==>}\; \Delta}$	$\dfrac{\Gamma \;\texttt{==>}\; \phi, \psi, \Delta}{\Gamma \;\texttt{==>}\; \phi \mid \psi, \Delta}$

Rules of Propositional Sequent Calculus

main	left side (work on antecedent)	right side (work on succedent)
not	$\dfrac{\Gamma \textbf{ ==> } \phi, \Delta}{\Gamma, \textbf{!}\phi \textbf{ ==> } \Delta}$	$\dfrac{\Gamma, \phi \textbf{ ==> } \Delta}{\Gamma \textbf{ ==> } \textbf{!}\phi, \Delta}$
and	$\dfrac{\Gamma, \phi, \psi \textbf{ ==> } \Delta}{\Gamma, \phi \,\&\, \psi \textbf{ ==> } \Delta}$	$\dfrac{\Gamma \textbf{ ==> } \phi, \Delta \quad \Gamma \textbf{ ==> } \psi, \Delta}{\Gamma \textbf{ ==> } \phi \,\&\, \psi, \Delta}$
or	$\dfrac{\Gamma, \phi \textbf{ ==> } \Delta \quad \Gamma, \psi \textbf{ ==> } \Delta}{\Gamma, \phi \,\vert\, \psi \textbf{ ==> } \Delta}$	$\dfrac{\Gamma \textbf{ ==> } \phi, \psi, \Delta}{\Gamma \textbf{ ==> } \phi \,\vert\, \psi, \Delta}$
imp	$\dfrac{\Gamma \textbf{ ==> } \phi, \Delta \quad \Gamma, \psi \textbf{ ==> } \Delta}{\Gamma, \phi \textbf{ -> } \psi \textbf{ ==> } \Delta}$	$\dfrac{\Gamma, \phi \textbf{ ==> } \psi, \Delta}{\Gamma \textbf{ ==> } \phi \textbf{ -> } \psi, \Delta}$

Rules of Propositional Sequent Calculus

main	left side (work on antecedent)	right side (work on succedent)
not	$\dfrac{\Gamma \text{ ==> } \phi, \Delta}{\Gamma, !\phi \text{ ==> } \Delta}$	$\dfrac{\Gamma, \phi \text{ ==> } \Delta}{\Gamma \text{ ==> } !\phi, \Delta}$
and	$\dfrac{\Gamma, \phi, \psi \text{ ==> } \Delta}{\Gamma, \phi \,\&\, \psi \text{ ==> } \Delta}$	$\dfrac{\Gamma \text{ ==> } \phi, \Delta \quad \Gamma \text{ ==> } \psi, \Delta}{\Gamma \text{ ==> } \phi \,\&\, \psi, \Delta}$
or	$\dfrac{\Gamma, \phi \text{ ==> } \Delta \quad \Gamma, \psi \text{ ==> } \Delta}{\Gamma, \phi \mid \psi \text{ ==> } \Delta}$	$\dfrac{\Gamma \text{ ==> } \phi, \psi, \Delta}{\Gamma \text{ ==> } \phi \mid \psi, \Delta}$
imp	$\dfrac{\Gamma \text{ ==> } \phi, \Delta \quad \Gamma, \psi \text{ ==> } \Delta}{\Gamma, \phi \text{ -> } \psi \text{ ==> } \Delta}$	$\dfrac{\Gamma, \phi \text{ ==> } \psi, \Delta}{\Gamma \text{ ==> } \phi \text{ -> } \psi, \Delta}$

CLOSE $\dfrac{}{\Gamma, \phi \text{ ==> } \phi, \Delta}$ **TRUE** $\dfrac{}{\Gamma \text{ ==> } \textbf{true}, \Delta}$ **FALSE** $\dfrac{}{\Gamma, \textbf{false} \text{ ==> } \Delta}$

Justification of Rules

Compute rules by applying semantics definition of connectives

Justification of Rules

Compute rules by applying semantics definition of connectives

$$\textbf{OR\_RIGHT}\ \frac{\Gamma \ \ \textbf{==>} \ \ \phi, \psi, \Delta}{\Gamma \ \ \textbf{==>} \ \ \phi \ \textbf{|} \ \psi, \Delta}$$

Follows directly from semantics of sequents

Justification of Rules

Compute rules by applying semantics definition of connectives

$$\textbf{OR\_RIGHT}\ \frac{\Gamma \ \textbf{==>}\ \phi, \psi, \Delta}{\Gamma \ \textbf{==>}\ \phi \ \textbf{|}\ \psi, \Delta}$$

Follows directly from semantics of sequents

$$\textbf{AND\_RIGHT}\ \frac{\Gamma \ \textbf{==>}\ \phi, \Delta \qquad \Gamma \ \textbf{==>}\ \psi, \Delta}{\Gamma \ \textbf{==>}\ \phi \ \textbf{\&}\ \psi, \Delta}$$

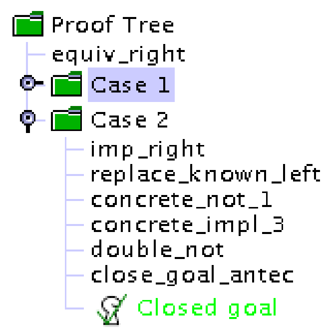
Γ **->** $(\phi$ **&** $\psi)$ **|** Δ **iff** Γ **->** ϕ **|** Δ **and** Γ **->** ψ **|** Δ

Distributivity of & over | and ->

Sequent Calculus Proofs

Goal to prove: $\mathcal{G} = \quad \psi_1, \ldots, \psi_m$ **==>** $\phi_1, \ldots, \phi_n$

- **find rule $\mathcal{R}$ whose conclusion matches $\mathcal{G}$**
- **instantiate $\mathcal{R}$ such that conclusion identical to $\mathcal{G}$**
- **recursively find proofs for resulting premisses $\mathcal{G}_1$, . . . , $\mathcal{G}_r$**
- **tree structure with goal as root**
- **close proof branch when rule without premise encountered**



Goal-directed proof search

In KeY tool proof displayed as JAVA Swing tree

A Simple Proof

$$\texttt{==>}\ (A \,\&\, (A \texttt{ -> } B)) \texttt{ -> } B$$

A Simple Proof

$$\overline{\qquad\qquad} \quad \overline{\qquad\qquad}$$

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$A \,\&\, (A \text{ -> } B) \text{ ==> } B$$

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$\text{==> } (A \,\&\, (A \text{ -> } B)) \text{ -> } B$$

By imp right $\dfrac{\Gamma, \phi \text{ ==> } \psi, \Delta}{\Gamma \text{ ==> } \phi \text{ -> } \psi, \Delta}$

A Simple Proof

$$\dfrac{\dfrac{\dfrac{\overline{} \quad \overline{}}{A, (A \text{ -> } B) \text{ ==> } B}}{A \,\&\, (A \text{ -> } B) \text{ ==> } B}}{\text{==> } (A \,\&\, (A \text{ -> } B)) \text{ -> } B}$$

By and left $\dfrac{\Gamma, \phi, \psi \text{ ==> } \Delta}{\Gamma, \phi \,\&\, \psi \text{ ==> } \Delta}$

A Simple Proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \texttt{ ==> } B, A} \qquad \dfrac{}{A, B \texttt{ ==> } B}}{A, (A \texttt{ -> } B) \texttt{ ==> } B}}{A \,\&\, (A \texttt{ -> } B) \texttt{ ==> } B}}{\texttt{==> } (A \,\&\, (A \texttt{ -> } B)) \texttt{ -> } B}$$

By imp left $\dfrac{\Gamma \texttt{ ==> } \phi, \Delta \qquad \Gamma, \psi \texttt{ ==> } \Delta}{\Gamma, \phi \texttt{ -> } \psi \texttt{ ==> } \Delta}$

A Simple Proof

$$\dfrac{\dfrac{\textsf{CLOSE}\dfrac{*}{A \;\textbf{==>}\; B, A} \qquad \dfrac{*}{A, B \;\textbf{==>}\; B}\textsf{CLOSE}}{A, (A \;\textbf{->}\; B) \;\textbf{==>}\; B}}{\dfrac{A \;\textbf{\&}\; (A \;\textbf{->}\; B) \;\textbf{==>}\; B}{\textbf{==>}\; (A \;\textbf{\&}\; (A \;\textbf{->}\; B)) \;\textbf{->}\; B}}$$

By close $\dfrac{}{\Gamma, \phi \;\textbf{==>}\; \phi, \Delta}$

A Simple Proof

$$\dfrac{\dfrac{\dfrac{\textsc{Close}\ \dfrac{*}{A \text{ ==> } B, A} \qquad \dfrac{*}{A, B \text{ ==> } B}\ \textsc{Close}}{A, (A \text{ -> } B) \text{ ==> } B}}{A \,\&\, (A \text{ -> } B) \text{ ==> } B}}{\text{==> } (A \,\&\, (A \text{ -> } B)) \text{ -> } B}$$

A proof is closed, if all its branches are closed.

Propositional Logic is insufficient

A ALL PERSONS ARE HAPPY

Propositional Logic is insufficient

A	**All persons are happy**
B	**Pat is a person**

Propositional Logic is insufficient

A	ALL PERSONS ARE HAPPY
B	PAT IS A PERSON
$?$	PAT IS HAPPY

Propositional logic lacks possibility to talk about individuals

In particular, need to model objects, attributes, associations, etc.

Propositional Logic is insufficient

A	ALL PERSONS ARE HAPPY
B	PAT IS A PERSON
$?$	PAT IS HAPPY

Propositional logic lacks possibility to talk about individuals

In particular, need to model objects, attributes, associations, etc.

$\Rightarrow$ **First-Order Logic (FOL)**