\[ \text{class} = 10 \land \text{ok} \land \text{wait} \]
\[ \text{class} = 10 \land \text{ok} \land \neg \text{wait} \]
Taking Stock

- We are playing a BIG formal game
Taking Stock

- We are playing a BIG formal game
- We have a well-defined extensible language:

  $$\text{Pred ::= true} \mid \ldots \mid \forall x \bullet P \mid \ldots$$
Taking Stock

- We are playing a BIG formal game
- We have a well-defined extensible language:
  \[
  \text{Pred ::= true} \mid \ldots \mid \forall x \cdot P \mid \ldots
  \]
- We have given it a well-defined meaning:
  \[
  \left[ \exists x \cdot P \right]_\rho \models \ldots
  \]
Taking Stock

- We are playing a BIG formal game
- We have a well-defined extensible language:
  \[ \text{Pred ::= true} \mid \ldots \mid \forall x \cdot P \mid \ldots \]

- We have given it a well-defined meaning:
  \[ \left[ \exists x \cdot P \right]_{\rho} \equiv \ldots \]

- We have provided rules to do proofs:
  \[
  F = G \\
  \frac{E[v := F] = E[v := G]}{E[v := F] = E[v := G]} \]
Taking Stock

- We are playing a BIG formal game
- We have a well-defined extensible language:

  \[ \text{Pred ::= } \text{true} | \ldots | \forall x \cdot P | \ldots \]

- We have given it a well-defined meaning:

  \[ [\exists x \cdot P]_\rho \equiv \ldots \]

- We have provided rules to do proofs:

  \[
  \begin{align*}
  F &= G \\
  E[v := F] &= E[v := G]
  \end{align*}
  \]

- Does it all make sense?
Truth and Provability

- **Statements**
  We are interested in statements of the following form:
  
  \[
  \text{Whenever predicates } P_1, \ldots, P_n \text{ are true, then so is } Q.
  \]

  (Let us use $\Gamma$ as a shorthand for $P_1, \ldots, P_n$).
Truth and Provability

- **Statements**
  We are interested in statements of the following form:

  \[ \text{Whenever predicates } P_1, \ldots, P_n \text{ are true, then so is } Q. \]

  (Let us use \( \Gamma \) as a shorthand for \( P_1, \ldots, P_n \)).

- **Truth** — \( \Gamma \models Q \)
  The above statement is *true* when, for every environment \( \rho \) that makes all the \( P_i \) true, we have \( \llbracket Q \rrbracket_{\rho} = \text{True} \). 

- **Provability** — \( \Gamma \vdash Q \)
  The above statement is *provable* when, given the \( P_i \) as assumptions, we can prove \( Q \) as a theorem.
Truth and Provability

- **Statements**
  We are interested in statements of the following form: 
  *Whenever predicates $P_1, \ldots, P_n$ are true, then so is $Q$.*
  (Let us use $\Gamma$ as a shorthand for $P_1, \ldots, P_n$).

- **Truth** — $\Gamma \models Q$
  The above statement is *true* when, for every environment $\rho$ that makes all the $P_i$ true, we have $\llbracket Q \rrbracket_\rho = \text{True}$.

- **Provability** — $\Gamma \vdash Q$
  The above statement is *provable* when, given the $P_i$ as assumptions, we can prove $Q$ as a theorem.
Truth vs. Provability

- Truth and Provability are *not* the same thing!
Truth vs. Provability

- Truth and Provability are *not* the same thing!

- **Soundness**
  A proof system is sound if whenever we can prove something, it is also true:

\[ \Gamma \vdash Q \text{ means that } \Gamma \models Q \]
Truth vs. Provability

- Truth and Provability are *not* the same thing!

- **Soundness**
  A proof system is sound if whenever we can prove something, it is also true:

  $$\Gamma \vdash Q \text{ means that } \Gamma \models Q$$

- **Completeness**
  A proof system is complete if whenever something is true, it can be proved to be so:

  $$\Gamma \models Q \text{ means that } \Gamma \vdash Q$$
Truth vs. Provability

- **Truth and Provability are not the same thing!**

- **Soundness**
  A proof system is sound if whenever we can prove something, it is also true:
  \[ \Gamma \vdash Q \text{ means that } \Gamma \models Q \]

- **Completeness**
  A proof system is complete if whenever something is true, it can proved to be so:
  \[ \Gamma \models Q \text{ means that } \Gamma \vdash Q \]

- The “Holy Grail” of formal systems is a sound and complete formal system
Soundness

- Soundness is a critical property
Soundness

- Soundness is a critical property
- Unsound proof systems allow “proofs” of false statements
  Very undesirable!
Soundness

- Soundness is a critical property
- Unsound proof systems allow “proofs” of false statements
  Very undesirable!
- Unsoundness can arise in one of two ways:
Soundness

- Soundness is a critical property
- Unsound proof systems allow “proofs” of false statements
  Very undesirable!
- Unsoundness can arise in one of two ways:
  - Inconsistency
Soundness

- Soundness is a critical property
- Unsound proof systems allow “proofs” of false statements
  Very undesirable!
- Unsoundness can arise in one of two ways:
  - **Inconsistency**
    - A proof system is inconsistent if anything can be proved using it
Soundness

- Soundness is a critical property
- Unsound proof systems allow “proofs” of false statements
  Very undesirable!
- Unsoundness can arise in one of two ways:
  - **Inconsistency**
    - A proof system is inconsistent if anything can be proved using it
    - In particular if we can prove *false*, we can prove anything.
Soundness

- Soundness is a critical property
- Unsound proof systems allow “proofs” of false statements
  Very undesirable!
- Unsoundness can arise in one of two ways:
  - **Inconsistency**
    - A proof system is inconsistent if anything can be proved using it
    - In particular if we can prove `false`, we can prove anything.
  - **Environment Mismatch** Even if consistent, a proof system can be unsound if our axioms and inference rules are incorrect, and fail to capture the truth properly.
Soundness

- Soundness is a critical property
- Unsound proof systems allow “proofs” of false statements
  Very undesirable!
- Unsoundness can arise in one of two ways:
  - **Inconsistency**
    - A proof system is inconsistent if anything can be proved using it
    - In particular if we can prove `false`, we can prove anything.
  - **Environment Mismatch** Even if consistent, a proof system can be unsound if our axioms and inference rules are incorrect, and fail to capture the truth properly.
- Ensuring soundness require great care in determining axioms and inference rules.
Completeness

- Completeness is a “nice-to-have” property
Completeness

- Completeness is a “nice-to-have” property
  - In principle a complete proof-system can be made fully automatic.
Completeness

- Completeness is a “nice-to-have” property
  - In principle a complete proof-system can be made fully automatic.

- Incompleteness simply means there are some true properties for which there are no proofs *in our formal system*. 

Gödel's Incompleteness Theorems (1931):

1. Any proof-system powerful enough for arithmetic cannot be both consistent and complete.
2. Any proof-system powerful enough to prove theorems about itself (arithmetic!), is inconsistent iff it can prove its own consistency.

Most formal systems we want to use embody arithmetic and so are incomplete as just described.
Completeness

- Completeness is a “nice-to-have” property
  - In principle a complete proof-system can be made fully automatic.
- Incompleteness simply means there are some true properties for which there are no proofs in our formal system.
- Gödel’s Incompleteness Theorems (1931):
Completeness

- Completeness is a “nice-to-have” property
  - In principle a complete proof-system can be made fully automatic.

- Incompleteness simply means there are some true properties for which there are no proofs *in our formal system*.

- Gödel’s Incompleteness Theorems (1931):
  1. Any proof-system powerful enough for arithmetic cannot be both consistent and complete.
Completeness

- Completeness is a “nice-to-have” property
  - In principle a complete proof-system can be made fully automatic.
- Incompleteness simply means there are some true properties for which there are no proofs in our formal system.
- Gödel’s Incompleteness Theorems (1931):
  1. Any proof-system powerful enough for arithmetic cannot be both consistent and complete.
  2. Any proof-system powerful enough to prove theorems about itself (arithmetic !), is inconsistent iff it can prove its own consistency.
Completeness

- Completeness is a “nice-to-have” property
  - In principle a complete proof-system can be made fully automatic.

- Incompleteness simply means there are some true properties for which there are no proofs *in our formal system*.

- Gödel’s Incompleteness Theorems (1931):
  1. Any proof-system powerful enough for arithmetic cannot be both consistent and complete.
  2. Any proof-system powerful enough to prove theorems about itself (arithmetic !), is inconsistent iff it can prove its own consistency

- Most formal systems we want to use embody arithmetic and so are incomplete as just described.
Meta-mathematics

- Meta-mathematics is the study of mathematics itself, as a “mathematical object”
Meta-mathematics

- Meta-mathematics is the study of mathematics itself, as a “mathematical object”
- A major focus of meta-mathematics is the study of proof-systems
Meta-mathematics

- Meta-mathematics is the study of mathematics itself, as a “mathematical object”
- A major focus of meta-mathematics is the study of proof-systems
- There is a large body of soundness and completeness results out there for different formal systems.
Meta-mathematics

- Meta-mathematics is the study of mathematics itself, as a “mathematical object”
- A major focus of meta-mathematics is the study of proof-systems
- There is a large body of soundness and completeness results out there for different formal systems.
- This course is based on a (mainly) sound but incomplete logic system
Meta-mathematics

- Meta-mathematics is the study of mathematics itself, as a “mathematical object”
- A major focus of meta-mathematics is the study of proof-systems
- There is a large body of soundness and completeness results out there for different formal systems.
- This course is based on a (mainly) sound but incomplete logic system
  - Mainly ???
Meta-mathematics

- Meta-mathematics is the study of mathematics itself, as a “mathematical object”
- A major focus of meta-mathematics is the study of proof-systems
- There is a large body of soundness and completeness results out there for different formal systems.
- This course is based on a (mainly) sound but incomplete logic system
  - Mainly ???
  - Yes, but …
Meta-mathematics

- Meta-mathematics is the study of mathematics itself, as a “mathematical object”
- A major focus of meta-mathematics is the study of proof-systems
- There is a large body of soundness and completeness results out there for different formal systems.
- This course is based on a (mainly) sound but incomplete logic system
  - Mainly ???
  - Yes, but …
  - We will be extending our language later by adding axioms of our own …
Formal Methods: current Research

GC6, Circus, Flash

- GC6 is a computing “grand-challenge” to develop libraries of verified software, currently with a number of “pilot projects”:
  - Mondex Smart Card (Natwest Bank)
  - POSIX filestore (NASA JPL)
- Circus is a formal language that combines imperative programming (variables, assignment) with concurrent systems (message-passing) (see [http://www.cs.york.ac.uk/circus/](http://www.cs.york.ac.uk/circus/)).
- work between TCD and York has focussed on formal models of Flash Memory
Handel-C, slotted-$\textit{Circus}$

- Handel-C is the C language extended with message-passing and parallelism, that compiles directly to hardware (FPGAs). (see http://www.agilityds.com/products/c_based_products/)
  - it is based on notion of synchronously clocked time-slots.
- “slotted-$\textit{Circus}$” is a generic framework for adding discrete time to $\textit{Circus}$, suitable for modelling time-slot languages like Handel-C.
  - it is currently be worked on here at TCD, using UTP as the semantic framework.
Real Life: BASE Trusted Gateway

Goal — Trusted Gateway for transferring messages between different security levels, for British Aerospace Systems & Equipment.
Real Life: BASE Trusted Gateway

**Goal** — Trusted Gateway for transferring messages between different security levels, for British Aerospace Systems & Equipment.

**Approach** — Two teams, one conventional, the other using formal methods.
Real Life: BASE Trusted Gateway

**Goal** — Trusted Gateway for transferring messages between different security levels, for British Aerospace Systems & Equipment.

**Approach** — Two teams, one conventional, the other using formal methods.

**Method** — Formal team employed VDM-SL, using IFAD Toolkit.
Real Life: BASE Trusted Gateway

Goal — Trusted Gateway for transferring messages between different security levels, for British Aerospace Systems & Equipment.

Approach — Two teams, one conventional, the other using formal methods.

Method — Formal team employed VDM-SL, using IFAD Toolkit.

BASE: Key Results (1)

- Formal approach spent more time up front in System Design (43% as against 34%).
BASE: Key Results (1)

- Formal approach spent more time up front in System Design (43% as against 34%).
- Formal approach uncovered an implicit special condition from requirements. Informal code had to be re-written at late stage to cope.
BASE: Key Results (1)

- Formal approach spent more time up front in System Design (43% as against 34%).
- Formal approach uncovered an implicit special condition from requirements. Informal code had to be re-written at late stage to cope.
- Formal code was less complex ("McCabe Complexity")
BASE: Key Results (1)

- Formal approach spent more time up front in System Design (43% as against 34%).
- Formal approach uncovered an implicit special condition from requirements. Informal code had to be re-written at late stage to cope.
- Formal code was less complex ("McCabe Complexity")
- Formal code one-fifth the size of informal code.
BASE: Key Results (2)

Formal system started up slower (4 times longer)

1. Formal System Invariant better understood, so more care was taken by resulting initialisation code.
BASE: Key Results (2)

Formal system started up slower (4 times longer)

1. Formal System Invariant better understood, so more care was taken by resulting initialisation code.
2. Not a big issue as the system is meant to stay up and running.
BASE: Key Results (3)

Formal system throughput higher
(almost 14 times faster)

1. The informal system had to have a last-minute fix, so the code speed got worse.
BASE: Key Results (3)

Formal system throughput higher (almost 14 times faster)

1. The informal system had to have a last-minute fix, so the code speed got worse.
2. If code is formally verified, then you don’t need so many run-time checks (array bounds, etc.)
class' = 10 \land ok' \land wait'
class' = 10 \land ok' \land \neg wait'