DTU Course 02156 Logical Systems and Logic Programming (2021)

Week	Date	Main Topics (Prolog Programming in All Lessons)
35 #01	31/8	Course Prerequisites & Tutorial on Logical Systems and Logic Programming
36 #02	7/9	Chapter 1 - Introduction (Prolog Note)
37 #03	14/9	Chapter 2 - Propositional Logic: Formulas, Models, Tableaux
38 #04	21/9	Chapter 3 - Propositional Logic: Deductive Systems
39 #05	28/9	"Isabelle" - Propositional Logic: Sequent Calculus Verifier (SeCaV)
40 #06	5/10	Chapter 4 - Propositional Logic: Resolution
41 #07	12/10	Chapter 7 - First-Order Logic: Formulas, Models, Tableaux
42		(Autumn Vacation)
43 #08	26/10	Chapter 8 - First-Order Logic: Deductive Systems
44 #09	2/11	"Isabelle" - First-Order Logic: Sequent Calculus Verifier (SeCaV)
45 #10	9/11	Chapter 9 - First-Order Logic: Terms and Normal Forms
46 #11	16/11	Chapter 10 - First-Order Logic: Resolution
47 #12	23/11	Chapter 11 - First-Order Logic: Logic Programming
48 #13	30/11	Chapter 12 - First-Order Logic: Undecidability and Model Theory & Course Evaluation

Responsible: Associate Professor Jørgen Villadsen <jovi@dtu.dk>

Assignments & Exam   MUST BE SOLVED INDIVIDUALLY

Assignment-1 Deadline Sunday 26/9 (Available Wednesday 15/9)

Assignment-2 Deadline Sunday 10/10 (Available Wednesday 29/9)

Assignment-3 Deadline Sunday 31/10 (Available Wednesday 13/10)

Assignment-4 Deadline Sunday 14/11 (Available Wednesday 3/11)

Assignment-5 Deadline Thursday 2/12 (Available Wednesday 17/11)

Written Exam Tuesday 14/12 (2 Hours / No Computer / All Notes Allowed)

The mandatory assignments and the written exam are evaluated as a whole – even if you do well in the mandatory assignments then you still must do decent in the written exam in order to pass the course!

A TEACHER MUST IMMEDIATELY REPORT ANY SUSPICION OF CHEATING TO THE STUDY ADMINISTRATION FOR FURTHER ACTIONS

Agenda — Week #8

Test

First-Order Logic (FOL) — Tableaux summary

FOL — The Gentzen System & The Hilbert System

More on Proofs — The Kepler Conjecture

Prolog note — Findall

Test

1. Is there a closed tableau for the formula $\neg p$?
2. Is the formula $\neg(p \wedge \neg p)$ valid?
3. Can a tableau branch if it is constructed using only α-rules?
4. Is the formula $\forall x \neg p(x)$ satisfiable?
5. Is $\vdash p \rightarrow (q \rightarrow p)$ an axiom in the Gentzen system $\mathcal{G}$?
6. Does there exist a proof of $(p \rightarrow q) \vee (q \rightarrow p)$ in the Hilbert system $\mathcal{H}$?

Tableaux Rules

α	α_1	α_2
$A_1 \wedge A_2$	A_1	A_2

β	β_1	β_2
$B_1 \vee B_2$	B_1	B_2

γ	$\gamma(a)$
$\forall x A(x)$	$A(a)$

δ	$\delta(a)$
$\exists x A(x)$	$A(a)$

The Gentzen System & The Hilbert System

A proof in the Gentzen system $\mathcal{G}$ is built from the closed tableau

1.	$\vdash \neg p, q, p$	Axiom
2.	$\vdash p \rightarrow q, p$	$\alpha \rightarrow$, 1
3.	$\vdash \neg p, p$	Axiom
4.	$\vdash \neg((p \rightarrow q) \rightarrow p), p$	$\beta \rightarrow$, 2, 3
5.	$\vdash ((p \rightarrow q) \rightarrow p) \rightarrow p$	$\alpha \rightarrow$, 4

The Gentzen System & The Hilbert System

A proof in the Gentzen system $\mathcal{G}$ is built from the closed tableau

1.	$\vdash \neg p, q, p$	Axiom
2.	$\vdash p \rightarrow q, p$	$\alpha \rightarrow$, 1
3.	$\vdash \neg p, p$	Axiom
4.	$\vdash \neg((p \rightarrow q) \rightarrow p), p$	$\beta \rightarrow$, 2, 3
5.	$\vdash ((p \rightarrow q) \rightarrow p) \rightarrow p$	$\alpha \rightarrow$, 4

A proof in the Hilbert system $\mathcal{H}$ usually requires more experience

1.	$(p \rightarrow q) \rightarrow p \vdash (p \rightarrow q) \rightarrow p$	Assumption
2.	$(p \rightarrow q) \rightarrow p \vdash \neg p \rightarrow (p \rightarrow q)$	Theorem 3.20
3.	$(p \rightarrow q) \rightarrow p \vdash \neg p \rightarrow p$	Transitivity 2, 1
4.	$(p \rightarrow q) \rightarrow p \vdash (\neg p \rightarrow p) \rightarrow p$	Theorem 3.31
5.	$(p \rightarrow q) \rightarrow p \vdash p$	MP 3, 4
6.	$\vdash ((p \rightarrow q) \rightarrow p) \rightarrow p$	Deduction 5

The Gentzen System I

Tableaux turned "tree upside down and signs reversed":

$\neg((p \vee q) \rightarrow (q \vee p))$

$p \vee q, \neg(q \vee p)$

$p \vee q, \neg q, \neg p$

$\overline{\text{L} \qquad \text{R}}$

$p, \neg q, \neg p$

$\times$

R

$q, \neg q, \neg p$

$\times$

1.	$\vdash \neg p, q, p$	Axiom
2.	$\vdash \neg q, q, p$	Axiom
3.	$\vdash \neg(p \vee q), q, p$	$\beta\vee$, 1, 2
4.	$\vdash \neg(p \vee q), (q \vee p)$	$\alpha\vee$, 3
5.	$\vdash (p \vee q) \rightarrow (q \vee p)$	$\alpha \rightarrow$, 4

$$\vdash (A \vee B) \rightarrow (B \vee A)$$

Similar for FOL

The Gentzen System I

Tableaux turned “tree upside down and signs reversed”:

$\neg((p \vee q) \rightarrow (q \vee p))$
$p \vee q, \neg(q \vee p)$
$p \vee q, \neg q, \neg p$

$\overline{\text{L} \qquad \text{R}}$

$p, \neg q, \neg p$
$\times$

R
$q, \neg q, \neg p$
$\times$

1.	$\vdash \neg p, q, p$	Axiom
2.	$\vdash \neg q, q, p$	Axiom
3.	$\vdash \neg(p \vee q), q, p$	$\beta\vee$, 1, 2
4.	$\vdash \neg(p \vee q), (q \vee p)$	$\alpha\vee$, 3
5.	$\vdash (p \vee q) \rightarrow (q \vee p)$	$\alpha \rightarrow$, 4

$$\vdash (A \vee B) \rightarrow (B \vee A)$$

Similar for FOL

But additional rules (δ and γ)

The Gentzen System I

Tableaux turned "tree upside down and signs reversed":

$\neg((p \vee q) \rightarrow (q \vee p))$
$p \vee q, \neg(q \vee p)$
$p \vee q, \neg q, \neg p$

$\overline{\text{L} \qquad \text{R}}$

L
$p, \neg q, \neg p$
$\times$

R
$q, \neg q, \neg p$
$\times$

1.	$\vdash \neg p, q, p$	Axiom
2.	$\vdash \neg q, q, p$	Axiom
3.	$\vdash \neg(p \vee q), q, p$	$\beta\vee$, 1, 2
4.	$\vdash \neg(p \vee q), (q \vee p)$	$\alpha\vee$, 3
5.	$\vdash (p \vee q) \rightarrow (q \vee p)$	$\alpha \rightarrow$, 4

$$\vdash (A \vee B) \rightarrow (B \vee A)$$

Similar for FOL

But additional rules (δ and γ)

Soundness and Completeness Theorem: $\models A$ **iff** $\vdash A$

The Gentzen System II

Tableaux:

$$
\begin{array}{c}
\underline{\neg\,((\forall x p(x) \lor \forall x q(x)) \to \forall x(p(x) \lor q(x)))} \\
\downarrow \\
\underline{\forall x p(x) \lor \forall x q(x)}, \neg \forall x(p(x) \lor q(x))
\end{array}
$$

$$
\begin{array}{cc}
\swarrow & \searrow \\
\forall x p(x), \underline{\neg \forall x(p(x) \lor q(x))} & \forall x q(x), \underline{\neg \forall x(p(x) \lor q(x))} \\
\downarrow & \downarrow \\
\forall x p(x), \underline{\neg\,(p(a) \lor q(a))} & \forall x q(x), \underline{\neg\,(p(a) \lor q(a))} \\
\downarrow & \downarrow \\
\underline{\forall x p(x)}, \neg p(a), \neg q(a) & \underline{\forall x q(x)}, \neg p(a), \neg q(a) \\
\downarrow & \downarrow \\
\forall x p(x), p(a), \neg p(a), \neg q(a) & \forall x q(x), q(a), \neg p(a), \neg q(a) \\
\times & \times
\end{array}
$$

The Gentzen System III

Proof:

$$
\begin{array}{cc}
\underline{\neg\forall x p(x), \neg p(a)}, p(a), q(a) & \underline{\neg\forall x q(x), \neg q(a)}, p(a), q(a) \\
\downarrow & \downarrow \\
\neg\forall x p(x), \underline{p(a), q(a)} & \neg\forall x q(x), \underline{p(a), q(a)} \\
\downarrow & \downarrow \\
\neg\forall x p(x), \underline{p(a) \vee q(a)} & \neg\forall x q(x), \underline{p(a) \vee q(a)} \\
\downarrow & \downarrow \\
\underline{\neg\forall x p(x)}, \forall x(p(x) \vee q(x)) & \underline{\neg\forall x q(x)}, \forall x(p(x) \vee q(x)) \\
\searrow & \swarrow \\
\end{array}
$$

$$
\begin{array}{c}
\underline{\neg\left(\forall x p(x) \vee \forall x q(x)\right), \forall x(p(x) \vee q(x))} \\
\downarrow \\
(\forall x p(x) \vee \forall x q(x)) \rightarrow \forall x(p(x) \vee q(x))
\end{array}
$$

The Hilbert System I

One rule MP: $\vdash A, \vdash A \rightarrow B \;/\; \vdash B$

The Hilbert System I

One rule MP: $\vdash A, \vdash A \to B \;/\; \vdash B$

Axiom 1: $\vdash A \to B \to A$

The Hilbert System I

One rule MP: $\vdash A, \vdash A \to B \ / \vdash B$

Axiom 1: $\vdash A \to B \to A$

Axiom 2: $\vdash (A \to B \to C) \to (A \to B) \to A \to C$

The Hilbert System I

One rule MP: $\vdash A, \vdash A \rightarrow B \;/\; \vdash B$

Axiom 1: $\vdash A \rightarrow B \rightarrow A$

Axiom 2: $\vdash (A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C$

Axiom 3: $\vdash (\neg B \rightarrow \neg A) \rightarrow A \rightarrow B$

The Hilbert System I

One rule MP: $\vdash A, \vdash A \rightarrow B \ / \vdash B$

Axiom 1: $\vdash A \rightarrow B \rightarrow A$

Axiom 2: $\vdash (A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C$

Axiom 3: $\vdash (\neg B \rightarrow \neg A) \rightarrow A \rightarrow B$

Complicated even for following tiny example.

1.	$\vdash (A \rightarrow (A \rightarrow A) \rightarrow A) \rightarrow (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$	Axiom 2
2.	$\vdash A \rightarrow (A \rightarrow A) \rightarrow A$	Axiom 1
3.	$\vdash (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$	MP 1, 2
4.	$\vdash A \rightarrow A \rightarrow A$	Axiom 1
5.	$\vdash A \rightarrow A$	MP 3, 4

The Hilbert System I

One rule MP: $\vdash A$, $\vdash A \to B \ / \vdash B$

Axiom 1: $\vdash A \to B \to A$

Axiom 2: $\vdash (A \to B \to C) \to (A \to B) \to A \to C$

Axiom 3: $\vdash (\neg B \to \neg A) \to A \to B$

Complicated even for following tiny example.

1.	$\vdash (A \to (A \to A) \to A) \to (A \to A \to A) \to A \to A$	Axiom 2
2.	$\vdash A \to (A \to A) \to A$	Axiom 1
3.	$\vdash (A \to A \to A) \to A \to A$	MP 1, 2
4.	$\vdash A \to A \to A$	Axiom 1
5.	$\vdash A \to A$	MP 3, 4

Similar for FOL

The Hilbert System I

One rule MP: $\vdash A$, $\vdash A \rightarrow B \ / \vdash B$

Axiom 1: $\vdash A \rightarrow B \rightarrow A$

Axiom 2: $\vdash (A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C$

Axiom 3: $\vdash (\neg B \rightarrow \neg A) \rightarrow A \rightarrow B$

Complicated even for following tiny example.

1.	$\vdash (A \rightarrow (A \rightarrow A) \rightarrow A) \rightarrow (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$	Axiom 2
2.	$\vdash A \rightarrow (A \rightarrow A) \rightarrow A$	Axiom 1
3.	$\vdash (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$	MP 1, 2
4.	$\vdash A \rightarrow A \rightarrow A$	Axiom 1
5.	$\vdash A \rightarrow A$	MP 3, 4

Similar for FOL

One additional rule and a few axioms follows

The Hilbert System II

Generalization rule in addition to MP:

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a)\ /\ \vdash \forall x A(x)$

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a) \ / \vdash \forall x A(x)$

Additional axioms:

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a) \;/ \vdash \forall x A(x)$

Additional axioms:

Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a) \;/\; \vdash \forall x A(x)$

Additional axioms:

Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

Axiom 5: $\vdash \forall x(A \rightarrow B(x)) \rightarrow (A \rightarrow \forall x B(x))$
provided that x is not free in A

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a) \;/\; \vdash \forall x A(x)$

Additional axioms:

Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

Axiom 5: $\vdash \forall x (A \rightarrow B(x)) \rightarrow (A \rightarrow \forall x B(x))$
provided that x is not free in A

Soundness and Completeness Theorem: $\models A$ **iff** $\vdash A$

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a) \ / \vdash \forall x A(x)$

Additional axioms:

Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

Axiom 5: $\vdash \forall x (A \rightarrow B(x)) \rightarrow (A \rightarrow \forall x B(x))$
provided that x is not free in A

Soundness and Completeness Theorem: $\models A$ **iff** $\vdash A$

Recall:

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a) \;/\; \vdash \forall x A(x)$

Additional axioms:

Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

Axiom 5: $\vdash \forall x (A \rightarrow B(x)) \rightarrow (A \rightarrow \forall x B(x))$
provided that x is not free in A

Soundness and Completeness Theorem: $\models A$ iff $\vdash A$

Recall:

Let U be a set of formulas and let $U \vdash A$ equal $\vdash A$ iff $U = \emptyset$

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a)$ / $\vdash \forall x A(x)$

Additional axioms:

Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

Axiom 5: $\vdash \forall x (A \rightarrow B(x)) \rightarrow (A \rightarrow \forall x B(x))$
provided that x is not free in A

Soundness and Completeness Theorem: $\models A$ **iff** $\vdash A$

Recall:

Let U be a set of formulas and let $U \vdash A$ equal $\vdash A$ iff $U = \emptyset$

Assumption rule: $U \vdash A_i$ $(A_i \in U)$

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a)$ / $\vdash \forall x A(x)$

Additional axioms:

Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

Axiom 5: $\vdash \forall x (A \rightarrow B(x)) \rightarrow (A \rightarrow \forall x B(x))$
provided that x is not free in A

Soundness and Completeness Theorem: $\models A$ **iff** $\vdash A$

Recall:

Let U be a set of formulas and let $U \vdash A$ equal $\vdash A$ iff $U = \emptyset$

Assumption rule: $U \vdash A_i$ $(A_i \in U)$

Deduction rule: $U \cup \{A\} \vdash B$ / $U \vdash A \rightarrow B$

The Hilbert System II

Generalization rule in addition to MP:

Rule Gen: $\vdash A(a)$ / $\vdash \forall x A(x)$

Additional axioms:

Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

Axiom 5: $\vdash \forall x(A \rightarrow B(x)) \rightarrow (A \rightarrow \forall x B(x))$
provided that x is not free in A

Soundness and Completeness Theorem: $\vDash A$ **iff** $\vdash A$

Recall:

Let U be a set of formulas and let $U \vdash A$ equal $\vdash A$ iff $U = \emptyset$

Assumption rule: $U \vdash A_i$ $(A_i \in U)$

Deduction rule: $U \cup \{A\} \vdash B$ / $U \vdash A \rightarrow B$

Deduction theorem: The deduction rule is a sound derived rule

The Hilbert System III

Rule Gen: $\vdash A(a) \;/\; \vdash \forall x A(x)$

The Hilbert System III

Rule Gen: $\vdash A(a) \;/\; \vdash \forall x A(x)$

In the presence of assumptions:

The Hilbert System III

Rule Gen: $\vdash A(a)$ / $\vdash \forall x A(x)$

In the presence of assumptions:

Rule Gen: $U \vdash A(a)$ / $U \vdash \forall x A(x)$
provided that a does not occur in U

The Hilbert System III

Rule Gen: $\vdash A(a)\ /\ \vdash \forall x A(x)$

In the presence of assumptions:

Rule Gen: $U \vdash A(a)\ /\ U \vdash \forall x A(x)$
provided that a does not occur in U

Without the proviso $A(a) \vdash \forall x A(x)$ would follow from $A(a) \vdash A(a)$ and since the Assumption Rule gives $A(a) \vdash A(a)$ then the Deduction Rule would give $\vdash A(a) \rightarrow \forall x A(x)$ which is not valid!

The Hilbert System III

Rule Gen: $\vdash A(a) \;/\; \vdash \forall x A(x)$

In the presence of assumptions:

Rule Gen: $U \vdash A(a) \;/\; U \vdash \forall x A(x)$
provided that a does not occur in U

Without the proviso $A(a) \vdash \forall x A(x)$ would follow from $A(a) \vdash A(a)$ and since the Assumption Rule gives $A(a) \vdash A(a)$ then the Deduction Rule would give $\vdash A(a) \rightarrow \forall x A(x)$ which is not valid!

Compare with Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

The Hilbert System III

Rule Gen: $\vdash A(a) \;/ \vdash \forall x A(x)$

In the presence of assumptions:

Rule Gen: $U \vdash A(a) \;/\; U \vdash \forall x A(x)$
provided that a does not occur in U

Without the proviso $A(a) \vdash \forall x A(x)$ would follow from $A(a) \vdash A(a)$ and since the Assumption Rule gives $A(a) \vdash A(a)$ then the Deduction Rule would give $\vdash A(a) \rightarrow \forall x A(x)$ which is not valid!

Compare with Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

Compare also with Theorem 8.14: $\vdash A(a) \rightarrow \exists x A(x)$

The Hilbert System III

Rule Gen: $\vdash A(a)$ / $\vdash \forall x A(x)$

In the presence of assumptions:

Rule Gen: $U \vdash A(a)$ / $U \vdash \forall x A(x)$
provided that a does not occur in U

Without the proviso $A(a) \vdash \forall x A(x)$ would follow from $A(a) \vdash A(a)$ and since the Assumption Rule gives $A(a) \vdash A(a)$ then the Deduction Rule would give $\vdash A(a) \rightarrow \forall x A(x)$ which is not valid!

Compare with Axiom 4: $\vdash \forall x A(x) \rightarrow A(a)$

Compare also with Theorem 8.14: $\vdash A(a) \rightarrow \exists x A(x)$

C-Rule: $U \vdash \exists x A(x)$ / $U \vdash A(a)$ "C" for "Choice"
provided that a does not occur in U or in $\exists x A(x)$ and Generalization is not used on a formula containing a constant introduced by the rule, each use of the rule introduces a new such constant and the final formula does not contain any such constants

The Flyspeck Project

The purpose of the flyspeck project is to produce a formal proof of the Kepler Conjecture.

The Flyspeck Project

The purpose of the flyspeck project is to produce a formal proof of the Kepler Conjecture.

Traditional mathematical proofs are written in a way to make them easily understood by mathematicians.

The Flyspeck Project

The purpose of the flyspeck project is to produce a formal proof of the Kepler Conjecture.

Traditional mathematical proofs are written in a way to make them easily understood by mathematicians.

Routine logical steps are omitted.

The Flyspeck Project

The purpose of the flyspeck project is to produce a formal proof of the Kepler Conjecture.

Traditional mathematical proofs are written in a way to make them easily understood by mathematicians.

Routine logical steps are omitted.

An enormous amount of context is assumed on the part of the reader.

The Flyspeck Project

The purpose of the flyspeck project is to produce a formal proof of the Kepler Conjecture.

Traditional mathematical proofs are written in a way to make them easily understood by mathematicians.

Routine logical steps are omitted.

An enormous amount of context is assumed on the part of the reader.

In a formal proof, all the intermediate logical steps are supplied.

The Flyspeck Project

The purpose of the flyspeck project is to produce a formal proof of the Kepler Conjecture.

Traditional mathematical proofs are written in a way to make them easily understood by mathematicians.

Routine logical steps are omitted.

An enormous amount of context is assumed on the part of the reader.

In a formal proof, all the intermediate logical steps are supplied.

No appeal is made to intuition, even if the translation from intuition to logic is routine.

The Flyspeck Project

The purpose of the flyspeck project is to produce a formal proof of the Kepler Conjecture.

Traditional mathematical proofs are written in a way to make them easily understood by mathematicians.

Routine logical steps are omitted.

An enormous amount of context is assumed on the part of the reader.

In a formal proof, all the intermediate logical steps are supplied.

No appeal is made to intuition, even if the translation from intuition to logic is routine.

https://code.google.com/archive/p/flyspeck/wikis/FlyspeckFactSheet.wiki

More on Proofs

Since 1930 it has been accepted that to any piece of mathematics expressed by a formula P there is a formal proof in FOL of a certain formula:

$$\vdash \mathit{Math} \rightarrow P$$

Math is a special fixed formula with the axioms of *set theory*, but the details belong to a course in the foundations of mathematics.

More on Proofs

Since 1930 it has been accepted that to any piece of mathematics expressed by a formula P there is a formal proof in FOL of a certain formula:

$$\vdash \mathit{Math} \rightarrow P$$

Math is a special fixed formula with the axioms of *set theory*, but the details belong to a course in the foundations of mathematics.

It could fit on a slide — but it would not be readily comprehensible.

More on Proofs

Since 1930 it has been accepted that to any piece of mathematics expressed by a formula P there is a formal proof in FOL of a certain formula:

$$\vdash \mathit{Math} \to P$$

Math is a special fixed formula with the axioms of *set theory*, but the details belong to a course in the foundations of mathematics.

It could fit on a slide — but it would not be readily comprehensible.

Usually the number 2 is coded as $\{\emptyset, \{\emptyset\}\}$.

More on Proofs

Since 1930 it has been accepted that to any piece of mathematics expressed by a formula P there is a formal proof in FOL of a certain formula:

$$\vdash \mathit{Math} \rightarrow P$$

Math is a special fixed formula with the axioms of *set theory*, but the details belong to a course in the foundations of mathematics.

It could fit on a slide — but it would not be readily comprehensible.

Usually the number 2 is coded as $\{\emptyset, \{\emptyset\}\}$.

In any case, the proof system $\vdash$ is part of the present course! :-)

More on Proofs

Since 1930 it has been accepted that to any piece of mathematics expressed by a formula P there is a formal proof in FOL of a certain formula:

$$\vdash \mathit{Math} \rightarrow P$$

Math is a special fixed formula with the axioms of *set theory*, but the details belong to a course in the foundations of mathematics.

It could fit on a slide — but it would not be readily comprehensible.

Usually the number 2 is coded as $\{\emptyset, \{\emptyset\}\}$.

In any case, the proof system $\vdash$ is part of the present course! :-)

Of course also the implication $\rightarrow$ and other operators.

More on Proofs

Since 1930 it has been accepted that to any piece of mathematics expressed by a formula P there is a formal proof in FOL of a certain formula:

$$\vdash \mathit{Math} \rightarrow P$$

Math is a special fixed formula with the axioms of *set theory*, but the details belong to a course in the foundations of mathematics.

It could fit on a slide — but it would not be readily comprehensible.

Usually the number 2 is coded as $\{\emptyset, \{\emptyset\}\}$.

In any case, the proof system $\vdash$ is part of the present course! :-)

Of course also the implication $\rightarrow$ and other operators.

Propositional logic is used as a "stepping stone" to FOL.

More on Proofs

Since 1930 it has been accepted that to any piece of mathematics expressed by a formula P there is a formal proof in FOL of a certain formula:

$$\vdash \mathit{Math} \to P$$

Math is a special fixed formula with the axioms of *set theory*, but the details belong to a course in the foundations of mathematics.

It could fit on a slide — but it would not be readily comprehensible.

Usually the number 2 is coded as $\{\emptyset, \{\emptyset\}\}$.

In any case, the proof system $\vdash$ is part of the present course! :-)

Of course also the implication $\to$ and other operators.

Propositional logic is used as a "stepping stone" to FOL.

Finally a proof system $\vdash$ for FOL will be implemented in Prolog.

More on Proofs

Since 1930 it has been accepted that to any piece of mathematics expressed by a formula P there is a formal proof in FOL of a certain formula:

$$\vdash Math \rightarrow P$$

Math is a special fixed formula with the axioms of *set theory*, but the details belong to a course in the foundations of mathematics.

It could fit on a slide — but it would not be readily comprehensible.

Usually the number 2 is coded as $\{\emptyset, \{\emptyset\}\}$.

In any case, the proof system $\vdash$ is part of the present course! :-)

Of course also the implication $\rightarrow$ and other operators.

Propositional logic is used as a “stepping stone” to FOL.

Finally a proof system $\vdash$ for FOL will be implemented in Prolog.

Implementations for propositional logic were part of the exercises.

The Kepler Conjecture — June 2010 Update

The name ‘flyspeck’ comes from matching the pattern /f.*p.*k/ against an English dictionary.

The Kepler Conjecture — June 2010 Update

The name ‘flyspeck’ comes from matching the pattern /f.*p.*k/ against an English dictionary.

FPK in turn is an acronym for ‘The Formal Proof of Kepler’ and the term ‘flyspeck’ can mean to examine closely or in minute detail; or to scrutinize.

The Kepler Conjecture — June 2010 Update

The name ‘flyspeck’ comes from matching the pattern /f.*p.*k/ against an English dictionary.

FPK in turn is an acronym for ‘The Formal Proof of Kepler’ and the term ‘flyspeck’ can mean to examine closely or in minute detail; or to scrutinize.

The term is thus quite appropriate for a project intended to scrutinize the minute details of a mathematical proof.

The Kepler Conjecture — June 2010 Update

The name ‘flyspeck’ comes from matching the pattern /f.*p.*k/ against an English dictionary.

FPK in turn is an acronym for ‘The Formal Proof of Kepler’ and the term ‘flyspeck’ can mean to examine closely or in minute detail; or to scrutinize.

The term is thus quite appropriate for a project intended to scrutinize the minute details of a mathematical proof.

The formal proof of the text part of the proof is estimated to be about 65% complete.

The Kepler Conjecture — June 2010 Update

The name ‘flyspeck’ comes from matching the pattern /f.*p.*k/ against an English dictionary.

FPK in turn is an acronym for ‘The Formal Proof of Kepler’ and the term ‘flyspeck’ can mean to examine closely or in minute detail; or to scrutinize.

The term is thus quite appropriate for a project intended to scrutinize the minute details of a mathematical proof.

The formal proof of the text part of the proof is estimated to be about 65% complete.

This is an undertaking of that has the potential to develop into one of historical proportions.

The Kepler Conjecture — June 2010 Update

The name ‘flyspeck’ comes from matching the pattern /f.*p.*k/ against an English dictionary.

FPK in turn is an acronym for ‘The Formal Proof of Kepler’ and the term ‘flyspeck’ can mean to examine closely or in minute detail; or to scrutinize.

The term is thus quite appropriate for a project intended to scrutinize the minute details of a mathematical proof.

The formal proof of the text part of the proof is estimated to be about 65% complete.

This is an undertaking of that has the potential to develop into one of historical proportions.

We are looking for mathematicians (from the advanced undergraduate level up) who are computer literate, and who are interested in transforming the way that mathematics is done.

The Kepler Conjecture — August 2014 Update

"We are pleased to announce the completion of the Flyspeck project, which has constructed a formal proof of the Kepler conjecture. The Kepler conjecture asserts that no packing of congruent balls in Euclidean 3-space has density greater than the face-centered cubic packing. It is the oldest problem in discrete geometry. The proof of the Kepler conjecture was first obtained by Ferguson and Hales in 1998. The proof relies on about 300 pages of text and on a large number of computer calculations."

The Kepler Conjecture — August 2014 Update

"We are pleased to announce the completion of the Flyspeck project, which has constructed a formal proof of the Kepler conjecture. The Kepler conjecture asserts that no packing of congruent balls in Euclidean 3-space has density greater than the face-centered cubic packing. It is the oldest problem in discrete geometry. The proof of the Kepler conjecture was first obtained by Ferguson and Hales in 1998. The proof relies on about 300 pages of text and on a large number of computer calculations."

For various reasons the Flyspeck Project uses type theory or Higher-Order Logic (HOL) instead of First-Order Logic (FOL), hence the formal proofs are different:

$$\vdash' \mathit{Math}' \rightarrow P'$$

In HOL the special fixed formula Math' is quite simple.

The Kepler Conjecture — August 2014 Update

"We are pleased to announce the completion of the Flyspeck project, which has constructed a formal proof of the Kepler conjecture. The Kepler conjecture asserts that no packing of congruent balls in Euclidean 3-space has density greater than the face-centered cubic packing. It is the oldest problem in discrete geometry. The proof of the Kepler conjecture was first obtained by Ferguson and Hales in 1998. The proof relies on about 300 pages of text and on a large number of computer calculations."

For various reasons the Flyspeck Project uses type theory or Higher-Order Logic (HOL) instead of First-Order Logic (FOL), hence the formal proofs are different:

$$\vdash' \mathit{Math}' \rightarrow P'$$

In HOL the special fixed formula Math' is quite simple.

The computer time used was around 8 Azure processor months...

Recall Map Colouring

Requirement: No two adjacent countries get the same colour
Use numbers 0,1,2,3 as colours

```
?- test(X).

X = [ (0, [1, 2, 1], austria),
      (0, [3, 2, 1], belgium),
      (0, [1], denmark),
      (3, [0, 1, 2, 0, 1], france),
      (1, [3, 0, 2, 2, 0, 0], germany),
      (2, [0, 1], holland),
      (1, [3, 0, 2], italy),
      (1, [0], portugal),
      (0, [3, 1], spain),
      (2, [3, 1, 0, 1], switzerland) ]

Yes
```

Findall Example

Prolog has a special *find-all* predicate:

```
findall(?Template,+Goal,?Bag)
```

It creates a list of the instantiations `Template` gets successively on backtracking over `Goal` and unifies the result with `Bag`.

```
solutions :-
  findall(X,test(X),L), length(L,N), write(N), nl.
```

Findall Example

Prolog has a special *find-all* predicate:

```
findall(?Template,+Goal,?Bag)
```

It creates a list of the instantiations `Template` gets successively on backtracking over `Goal` and unifies the result with `Bag`.

```
solutions :-
  findall(X,test(X),L), length(L,N), write(N), nl.
```

In this case `bagof` and `setof` would give the same result:

```
?- solutions.
7776

Yes
```

The Standard Order of Terms

Prolog has a so-called standard order of terms:

$$\text{Variables} < \text{Numbers} < \text{Atoms} < \text{Compound Terms}$$

The order of variables is system dependent, but in SWI-Prolog the variables are ordered by their address (the oldest variable first)

The Standard Order of Terms

Prolog has a so-called standard order of terms:

$$\text{Variables} < \text{Numbers} < \text{Atoms} < \text{Compound Terms}$$

The order of variables is system dependent, but in SWI-Prolog the variables are ordered by their address (the oldest variable first)

Numbers are ordered by value and atoms alphabetically

The Standard Order of Terms

Prolog has a so-called standard order of terms:

$$\text{Variables} < \text{Numbers} < \text{Atoms} < \text{Compound Terms}$$

The order of variables is system dependent, but in SWI-Prolog the variables are ordered by their address (the oldest variable first)

Numbers are ordered by value and atoms alphabetically

Compound terms are first ordered on their arity, then alphabetically on their functor and then finally on their arguments (the leftmost argument first)

The Standard Order of Terms

Prolog has a so-called standard order of terms:

$$\text{Variables} < \text{Numbers} < \text{Atoms} < \text{Compound Terms}$$

The order of variables is system dependent, but in SWI-Prolog the variables are ordered by their address (the oldest variable first)

Numbers are ordered by value and atoms alphabetically

Compound terms are first ordered on their arity, then alphabetically on their functor and then finally on their arguments (the leftmost argument first)

A list is treated as a compound term whose functor is `./2`

The Standard Order of Terms

Prolog has a so-called standard order of terms:

$$\text{Variables} < \text{Numbers} < \text{Atoms} < \text{Compound Terms}$$

The order of variables is system dependent, but in SWI-Prolog the variables are ordered by their address (the oldest variable first)

Numbers are ordered by value and atoms alphabetically

Compound terms are first ordered on their arity, then alphabetically on their functor and then finally on their arguments (the leftmost argument first)

A list is treated as a compound term whose functor is `./2`

`Term1 @< Term2` succeeds iff `Term1` $<$ `Term2`
`Term1 @=< Term2` succeeds iff `Term1` $\leq$ `Term2`
`Term1 @> Term2` succeeds iff `Term1` $>$ `Term2`
`Term1 @>= Term2` succeeds iff `Term1` $\geq$ `Term2`

The Standard Order of Terms — Examples

```
?- a @< a.        ?- a @< b.        ?- one @< 2.

No                Yes               No
```

The Standard Order of Terms — Examples

```
?- a @< a.

No
```

```
?- a @< b.

Yes
```

```
?- one @< 2.

No
```

```
?- X = a, Y = b, X @< Y.

X = a
Y = b

Yes
```

```
?- a @< V.

No
```

```
?- Z @< V.

Yes
```

The Standard Order of Terms — Examples

```
?- a @< a.

No
```

```
?- a @< b.

Yes
```

```
?- one @< 2.

No
```

```
?- X = a, Y = b, X @< Y.

X = a
Y = b

Yes
```

```
?- a @< V.

No
```

```
?- Z @< V.

Yes
```

```
?- X = X, Y = Y, X @< Y.

Yes
```

```
?- Y = Y, X = X, X @< Y.

No
```

More on Standard Order of Terms

SWI-Prolog:

```
y :- X @< Y, f(X) = f(X), f(Y) = f(Y), X @< Y.

n :- X @< Y, f(Y) = f(Y), f(X) = f(X), X @< Y.

?- y.

Yes

?- n.

No
```

More on Standard Order of Terms

SWI-Prolog:

```
y :- X @< Y, f(X) = f(X), f(Y) = f(Y), X @< Y.

n :- X @< Y, f(Y) = f(Y), f(X) = f(X), X @< Y.

?- y.

Yes

?- n.

No
```

YAP-Prolog: Both succeeds!

More on Standard Order of Terms

SWI-Prolog:

```
y :- X @< Y, f(X) = f(X), f(Y) = f(Y), X @< Y.

n :- X @< Y, f(Y) = f(Y), f(X) = f(X), X @< Y.

?- y.

Yes

?- n.

No
```

YAP-Prolog: Both succeeds!

GNU-Prolog: Both fails!

All Solutions — Example

```
man(socrates). man(plato).

mortal(X) :- man(X). mortal(socrates). mortal(fido).

?- mortal(X).

X = socrates ;

X = plato ;

X = socrates ;

X = fido ;

No
```

All Solutions — Motivation

The following query recomputes the first solution to get the second solution and besides it does not generalize to many solutions.

```
man(socrates). man(plato).

mortal(X) :- man(X). mortal(socrates). mortal(fido).

?- mortal(X), mortal(Y), X \= Y.

X = socrates
Y = plato

Yes
```

How to obtain a list of all solutions to a goal?

Should work for any goal and be efficient.

All Solutions — Definition

Prolog has a special *find-all* predicate:

```
findall(?Template,+Goal,?Bag)
```

It creates a list of the instantiations `Template` gets successively on backtracking over `Goal` and unifies the result with `Bag`.

```
man(socrates). man(plato).

mortal(X) :- man(X). mortal(socrates). mortal(fido).

?- findall(X,mortal(X),L).

L = [socrates, plato, socrates, fido] ;

No
```

Findall, Bagof and Setof

```
?- findall(X,member((_,X),[(1,a),(1,a),(2,c),(2,b)]),S).

S = [a, a, c, b] ;

No
```

There are 2 alternative special predicates:

```
?- bagof(...).

S = [a, a] ;

S = [c, b] ;

No
```

```
?- setof(...).

S = [a] ;

S = [b, c] ;

No
```

Findall Never Fails

```
?- findall(_,fail,S).

S = []

Yes

?- bagof(_,fail,S).

No
```

In general `findall` should be used.

Findall Never Fails

```
?- findall(_,fail,S).

S = []

Yes

?- bagof(_,fail,S).

No
```

In general `findall` should be used.

There are many other features of `findall`, `bagof` and `setof` that cannot be explained here.

Findall Never Fails

```
?- findall(_,fail,S).

S = []

Yes

?- bagof(_,fail,S).

No
```

In general `findall` should be used.

There are many other features of `findall`, `bagof` and `setof` that cannot be explained here.

But be careful not to loop: `?- findall(_,repeat,_).`