A Short Course on Program Extraction

Monika Seisenberger

Swansea University

Verona, 26-29 May 2020

Aims of the course

Main question of the course:

> *What is the computational content behind a proof?*

Aims of the course

Main question of the course:

> *What is the computational content behind a proof?*

To this end, we will demonstrate that logic is a natural bridge between mathematics and computation.

Aims of the course

Main question of the course:

What is the computational content behind a proof?

To this end, we will demonstrate that logic is a natural bridge between mathematics and computation.

We will study how valid reasoning in abstract mathematics leads to provably correct algorithms and hence certified computer programs.

What do we need for this endeavor?

1) A proof calculus (or in general the abilty to express statements in logic and to do proofs).

What do we need for this endeavor?

1) A proof calculus (or in general the abilty to express statements in logic and to do proofs).

2) A tool that can carry out the proofs for us.

What do we need for this endeavor?

1) A proof calculus (or in general the abilty to express statements in logic and to do proofs).

2) A tool that can carry out the proofs for us.

3) A way to store the proofs.

What do we need for this endeavor?

1) A proof calculus (or in general the abilty to express statements in logic and to do proofs).

2) A tool that can carry out the proofs for us.

3) A way to store the proofs.

4) A mechanism to extract the computational content from a proof.

What do we need for this endeavor? (2)

1) A proof calculus (or in general the abilty to express statements in logic and to do proofs). [Natural deduction]

2) A tool that can carry out the proofs for us. [The Minlog proof system]

3) A way to write up and store the proofs. [λ*calculus*]

4) A mechanism to extract the computational content from a proof. [Realizability]

5) Interesting case studies and applications.

Outline

The fundamental idea of program extraction

The fundamental idea of program extraction

A *proof* is a construction, represented by a text or a finite tree, that convinces us that a formula is *true*.

The fundamental idea of program extraction

A *proof* is a construction, represented by a text or a finite tree, that convinces us that a formula is *true*.

Often, a formula can also be understood as a *computational problem*.

The fundamental idea of program extraction

A *proof* is a construction, represented by a text or a finite tree, that convinces us that a formula is *true*.

Often, a formula can also be understood as a *computational problem*.

For example, the formula stating that there are infinitely many prime numbers,

$$\forall x\, \exists y\, (y > x \wedge \mathbf{Prime}(y))$$

can be understood as the problem of computing for every natural number x a prime number y that is greater than x.

The fundamental idea of program extraction

A *proof* is a construction, represented by a text or a finite tree, that convinces us that a formula is *true*.

Often, a formula can also be understood as a *computational problem*.

For example, the formula stating that there are infinitely many prime numbers,

$$\forall x\,\exists y\,(y > x \land \mathbf{Prime}(y))$$

can be understood as the problem of computing for every natural number x a prime number y that is greater than x.

Program extraction is based on the observation that a proof not only represents an argument why a formula is true but also contains a program that solves the computational problem it expresses.

Predicate logic (a.k.a. first-order logic, FOL)

Gottlob Frege (1848 - 1925)

Predicate logic was introduced by Frege in his *Begriffsschrift*.

The language of predicate logic

The language of predicate logic

Example: “Every positive number has a positive square root”

The language of predicate logic

Example: "Every positive number has a positive square root"

$$\forall x\,(x > 0 \to \exists y\,(y > 0 \land x = y * y))$$

The language of predicate logic

Example: "Every positive number has a positive square root"

$$\forall x\,(x > 0 \to \exists y\,(y > 0 \land x = y * y))$$

The *language*, $\mathcal{L} = (\mathcal{C}, \mathcal{F}, \mathcal{P})$, for this formula consists of

Constants: $\mathcal{C} = \{0\}$

Function symbols: $\mathcal{F} = \{*\}$

Predicate symbols: $\mathcal{P} = \{>\}$

The language of predicate logic

Example: “Every positive number has a positive square root”

$$\forall x\,(x > 0 \to \exists y\,(y > 0 \land x = y * y))$$

The *language*, $\mathcal{L} = (\mathcal{C}, \mathcal{F}, \mathcal{P})$, for this formula consists of

Constants: $\mathcal{C} = \{0\}$

Function symbols: $\mathcal{F} = \{*\}$

Predicate symbols: $\mathcal{P} = \{>\}$

The elements of $\mathcal{L}$ are also called *non-logical symbols*. The choice of $\mathcal{L}$ may vary depending on the intended application.

The language of predicate logic

Example: "Every positive number has a positive square root"

$$\forall x\,(x > 0 \to \exists y\,(y > 0 \land x = y * y))$$

The *language*, $\mathcal{L} = (\mathcal{C}, \mathcal{F}, \mathcal{P})$, for this formula consists of

Constants: $\mathcal{C} = \{0\}$
Function symbols: $\mathcal{F} = \{*\}$
Predicate symbols: $\mathcal{P} = \{>\}$

The elements of $\mathcal{L}$ are also called *non-logical symbols*. The choice of $\mathcal{L}$ may vary depending on the intended application.

The other symbols occurring in a formula of predicate logic are application independent and are called *logical symbols*:

Variables: $x, y, \ldots$
Logical constants: $\top$ ("true"), '$\bot$" (false)
Logical connectives: $\land$ ("and"), $\lor$ ("or"), $\to$ ("implies")
Quantifiers: $\forall$ ("for all"), $\exists$ ("exists")
Equality: $=$

Negation can be defined as $\neg A \stackrel{\text{Def}}{=} A \to \bot$.

The semantics of predicate logic

Alfred Tarski (1901-1983)

Tarski was the first to systematically study the notion of truth for formulas in predicate logic.

Models

A *model* (or *structure*) $\mathcal{M}$ for a language $\mathcal{L} = (\mathcal{C}, \mathcal{F}, \mathcal{P})$ consists of:

- a nonempty set M, called the *carrier set of* $\mathcal{M}$
- an interpretation in M of
  - the constants in $\mathcal{C}$,
  - the function symbols in $\mathcal{F}$,
  - the predicate symbols in $\mathcal{P}$.

In a given model $\mathcal{M}$, any $\mathcal{L}$-formula is either true or false.

Truth, Validity, Logical Consequence

$\mathcal{M} \models A$ (formula A is *true* in model $\mathcal{M}$, or $\mathcal{M}$ *satisfies* A)

A is *logically valid* ($\models A$) $\stackrel{\text{Def}}{=}$ for *all* models $\mathcal{M}$, $\mathcal{M} \models A$.

(A is true in *all* models)

A is a *logical consequence* of a set of formulas Γ ($\Gamma \models A$) $\stackrel{\text{Def}}{=}$ for *all* models $\mathcal{M}$, if $\mathcal{M} \models \Gamma$, then $\mathcal{M} \models A$.

(A is true in *all* models of Γ, or Γ logically implies A)

Where $\mathcal{M} \models \Gamma$ means $\mathcal{M} \models B$ for all $B \in \Gamma$.

Proofs

A *proof system* is a collection of rules to derive logically valid formulas.

There are many different proof systems. A popular, due to Gentzen, is called *Natural Deduction* since its rules are close to natural human reasoning.

Gerhard Gentzen (1909 - 1945)

Natural Deduction

Assumption rule u:A (assumptions are cancelled by $\to^+ u : A$)		
	Introduction rules	Elimination rules
$\wedge$	$\dfrac{A \qquad B}{A \wedge B}\ \wedge^+$	$\dfrac{A \wedge B}{A}\ \wedge^-_{\mathrm{l}} \qquad \dfrac{A \wedge B}{B}\ \wedge^-_{\mathrm{r}}$
$\to$	$\dfrac{B}{A \to B}\ \to^+ u : A$	$\dfrac{A \to B \qquad A}{B}\ \to^-$
$\vee$	$\dfrac{A}{A \vee B}\ \vee^+_{\mathrm{l}} \qquad \dfrac{B}{A \vee B}\ \vee^+_{\mathrm{r}}$	$\dfrac{A \vee B \qquad A \to C \qquad B \to C}{C}\ \vee^-$
$\bot$		$\dfrac{\bot}{A}\ \mathrm{efq} \qquad \dfrac{\neg\neg A}{A}\ \mathrm{raa}$

Quantifier rules

	Introduction rules	Elimination rules
$\forall$	$\dfrac{A(x)}{\forall x\, A(x)}\ \forall^+ \quad (*)$	$\dfrac{\forall x\, A(x)}{A(t)}\ \forall^-$
$\exists$	$\dfrac{A(t)}{\exists x\, A(x)}\ \exists^+$	$\dfrac{\exists x\, A(x) \qquad \forall x\, (A(x) \to C)}{C}\ \exists^- \quad (**)$

Variable conditions:

(*) x must not occur free in any free (that is, uncancelled) assumption.

(**) x must not occur free in C.

Adding these rules to natural deduction yields a complete proof system.

Natural Deduction (version with explicit assumptions)

Assumption rule $\dfrac{}{\Gamma, A \vdash A}$ use

	Introduction rules	Elimination rules
$\wedge$	$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge^+$	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge^-_{\mathsf{l}} \qquad \dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge^-_{\mathsf{r}}$
$\to$	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \to^+$	$\dfrac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \to^-$
$\vee$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee^+_{\mathsf{l}} \qquad \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee^+_{\mathsf{r}}$	$\dfrac{\Gamma \vdash A \vee B \qquad \Gamma \vdash A \to C \qquad \Gamma \vdash B \to C}{\Gamma \vdash C}\ \vee^-$
$\bot$		$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \mathsf{efq} \qquad \dfrac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}\ \mathsf{raa}$
$\forall$	$\dfrac{\Gamma \vdash A(x)}{\Gamma \vdash \forall x\, A(x)}\ \forall^+$ (x not free in Γ)	$\dfrac{\Gamma \vdash \forall x\, A(x)}{\Gamma \vdash A(t)}\ \forall^-$
$\exists$	$\dfrac{\Gamma \vdash A(t)}{\Gamma \vdash \exists x\, A(x)}\ \exists^+$	$\dfrac{\Gamma \vdash \exists x\, A(x) \qquad \Gamma \vdash \forall x\, (A(x) \to C)}{\Gamma \vdash C}\ \exists^-$ (x not free in Γ, C)

Equality rules (both versions)

	Introduction rule	Elimination rule
$=$	$\dfrac{}{t = t}$	$\dfrac{A(s) \qquad s = t}{A(t)}$

	Introduction rule	Elimination rule
$=$	$\dfrac{}{\Gamma \vdash t = t}$	$\dfrac{\Gamma \vdash A(s) \qquad \Gamma \vdash s = t}{\Gamma \vdash A(t)}$

Symmetry and transitivity of equality can be derived from these rules.

Example: Equivalence of $A \to (B \to C)$ and $A \wedge B \to C$

$$
\dfrac{\dfrac{\dfrac{\dfrac{u : A \wedge B \to C \qquad \dfrac{v : A \qquad w : B}{A \wedge B}\,\wedge^+}{C}\,\to^-}{B \to C}\,\to^+ w : B}{A \to (B \to C)}\,\to^+ v : A}{(A \wedge B \to C) \to (A \to (B \to C))}\,\to^+ u : A \wedge B \to C
$$

Exercise: Prove other direction on your own.

The Brouwer-Heyting-Kolmogorov Interpretation:

According to the BHK interpretation a formula expresses a *computational problem* which is defined by a description of how to solve it:

A solution to $A \wedge B$ is a pair (a, b) such that

$$a \text{ solves } A \text{ and } b \text{ solves } B.$$

The Brouwer-Heyting-Kolmogorov Interpretation:

According to the BHK interpretation a formula expresses a *computational problem* which is defined by a description of how to solve it:

A solution to $A \wedge B$ is a pair (a, b) such that

a solves A and b solves B.

A solution to $A \vee B$ is

either $(0, a)$ where a solves A

or $(1, b)$ where b solves B.

A solution to $A \to B$ is a construction that transforms

any solution of A to a solution of B.

The lambda calculus

In the BHK interpretation it is left open what a "construction" is.

Church's *lambda calculus* provides a good notion of construction:

The lambda calculus consists of

- *lambda terms* generated by the rules

x	Variables
$\lambda x \,.\, M$	lambda-abstraction
$M\,N$	Application

- *beta-reduction*

$$(\lambda x \,.\, M)N \quad \to_\beta \quad M[N/x]$$

$M[N/x]$ denotes substitution of the term N for x in the term M.

One usually writes $M\,N\,K$ for $(M\,N)\,K$.

Lambda calculus with types

Types are like propositional formulas with $\to$ as the only connective.

Let $\Gamma = x_1 : A_1, \dots x_n, : A_n$ be a *context*, that is a type assignment to variables.

We define inductively the relation $\Gamma \vdash M : A$
(M has type A in context Γ).

$$\Gamma, x : A \vdash x : A$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x\, M : A \to B} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : B}$$

β-reduction and -equality

Theorem

β-reduction is *strongly normalizing*, that is, every reduction sequence $M_1 \to_\beta M_2 \to_\beta M_3, \ldots$ terminates.

Theorem

β-reduction is *confluent*, that is, if $M \to^*_\beta N_1$ and $M \to^*_\beta N_2$, then there exists a term N such that $N_1 \to^*_\beta N$ and $N_2 \to^*_\beta N$.

Theorem

The relation of β-equality, defined by

$$M =_\beta N \quad :\Leftrightarrow \quad \exists K \, (M \to^*_\beta K \wedge N \to^*_\beta K)$$

is decidable.

Extension to products and sums

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \times B}$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_0(M) : A} \qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_1(M) : B}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash (0, M) : A + B} \qquad \frac{\Gamma \vdash M : B}{\Gamma \vdash (1, M) : A + B}$$

$$\frac{\Gamma \vdash M : A + B \qquad \Gamma \vdash N : A \to C \qquad \Gamma \vdash K : B \to C}{\Gamma \vdash \mathbf{case}(M, N, K)}$$

$$\begin{array}{lll}
\pi_0(M, N) & \to_\beta & M \\
\pi_1(M, N) & \to_\beta & N \\
\mathbf{case}((0, M), N, K) & \to_\beta & N\, M \\
\mathbf{case}((1, M), N, K) & \to_\beta & K\, M
\end{array}$$

The Curry-Howard correspondence

The *Curry-Howard correspondence* is the observation that intuitionistic natural deduction proofs are in a natural correspondence with the typed lambda calculus.

Since typed lambda terms are the core of functional programming languages such as ML and Haskell (named after Haskell B Curry) one can also say that intuitionistic proofs correspond to programs.

Haskell B Curry (1900-1982)

Intuitionistic ND proofs vs typed lambda calculus

$\dfrac{A \qquad B}{A \wedge B}$	$\dfrac{M : A \qquad N : B}{(M, N) : A \times B}$
$\dfrac{A \wedge B}{A} \qquad \dfrac{A \wedge B}{B}$	$\dfrac{M : A \times B}{\pi_0(M) : A} \qquad \dfrac{M : A \times B}{\pi_1(M) : B}$
$\dfrac{B}{A \to B} \to^+ u : A$	$\dfrac{M : B}{\lambda x\, M : A \to B}$
$\dfrac{A \to B \qquad A}{B}$	$\dfrac{M : A \to B \qquad N : B}{M\, N : B}$
$\dfrac{A}{A \vee B} \qquad \dfrac{B}{A \vee B}$	$\dfrac{M : A}{(0, M) : A \vee B} \qquad \dfrac{M : B}{(1, M) : A \vee B}$
$\dfrac{A \vee B \qquad A \to C \qquad B \to C}{C}$	$\dfrac{M : A \vee B \qquad N : A \to C \qquad K : B \to C}{\mathbf{case}(M, N, K) : C}$