

CL 2002:

Computational Logic

(Lecture 1)

Sergei N. Artemov

September 3, 2002

*Computer Science Program
CUNY Graduate Center*

TIME100 project: "The greatest scientists of the century"
(20 positions):

- Technology - 6 (airplane, rocket, TV, transistor, plastic, WWW)
- Biology & Medicine - 4 (psychoanalysis, penicillin, DNA, polio)
- Physics & Astronomy - 3 (Einstein, Fermi, Hubble)
- Anthropology - 1 (The Leakeys)
- Economy - 1 (Keynes)
- Environment - 1 (Rachel Carson)
- Psychology - 1 (Piaget)

- Computer Science - 1 (?)
- Mathematics - 1 (?)
- Philosophy - 1 (?)

TIME100 project: "The greatest scientists of the century"
(20 positions):

- Technology - 6 (airplane, rocket, TV, transistor, plastic, WWW)
- Biology & Medicine - 4 (psychoanalysis, penicillin, DNA, polio)
- Physics & Astronomy - 3 (Einstein, Fermi, Hubble)
- Anthropology - 1 (The Leakeys)
- Economy - 1 (Keynes)
- Environment - 1 (Rachel Carson)
- Psychology - 1 (Piaget)

- Computer Science - 1 (Turing, a logician)
- Mathematics - 1 (Gödel, a logician)
- Philosophy - 1 (Wittgenstein, who began as a logician)

Three traditions in Logic

1. *Classical*: Frege, Hilbert, Gödel, Tarski
2. *Constructive*: Brouwer, Heyting, Kolmogorov, Gödel
3. *Explicit*: Skolem, Curry, Gödel, Church - *bridge to computing!*

Fundamental results in Logic make their way from foundational topics to real applications within one lifetime

Propositional Logic: consistent, decidable, not feasible (if $P \neq NP$). Many practical methods reduce to this level (Deep Space 1)

Boole (1854), Post (1920): boolean circuits, boolean values in programming, verification (Shuttle example), duality of proof search and model checking

Gentzen (1933): normalization of proofs basis for "all" modern provers, proof checkers

Downside: limited expressive power!

First order logic: one sort of objects, quantifiers \forall, \exists . Consistent, undecidable but recursively enumerable, formalizes all "usual" mathematical reasoning. First order theories are typically not categorical. Formal arithmetic, set theory, theory of reals, theory of groups, rings, fields, etc. can be presented as FO theories.

Frege (1879): proofs in first order systems as positive tests of validity

Downside: not "object oriented", no direct representation of dynamic features (time, actions, etc.) hence excessive coding, computationally unfriendly! Another trouble: \exists -sickness (later).

Logic Programming: computing by proving

J. Robinson (1965): resolution + unification = single rule proof system for the first order logic. A theoretical prototype of PROLOG programming language (1972).

Verification problem: "does a program satisfy its specification?" is subsumed by the programming language. Japanese "fifth generation computer initiative". Pure PROLOG: easy to write a program, "almost" the usual math slang. The language of choice in AI (John McCarthy CS Colloquium talk at GC on Thursday September 26.)

Downside: Relatively slow. Later refinements improved its efficiency but the logical purity has been lost.

Higher order logic: Higher sorts of quantified variables. "Very" undecidable, not axiomatizable, not compact. HO theories: second order arithmetic (analysis), type theories. Close to the natural language, widely used for proof checking in verification.

Andrews (1982): classical HOL prover

Downside: No efficient proof search, no complete proof systems possible, difficulties with semantics and consistency.

Modal Logic: (Lewis & Langford (1932)) extends the usual logic (typically propositional) by new atoms $\Box F$ - "F is necessary" to capture dynamics. Usually preserves decidability

Gödel (1933): provability calculus (= **S4**). The most popular modal logic: knowledge representation, dynamic logic, nonmonotonic reasoning, etc.

1. Classical axioms and rules
2. $\Box(F \rightarrow G) \rightarrow (\Box F \rightarrow \Box G)$
3. $\Box F \rightarrow F$
4. $\Box F \rightarrow \Box \Box F$
5. $F / \Box F$ (necessitation rule)

Gödel's provability problem: find an intended provability model for **S4**. Remained open for > 60 years.

McKinsey - Tarski (1948): topological semantics $\Box F = \text{interior}(F)$,
leads to logics for dynamic systems

Kripke (1959): possible worlds à la Leibniz

Hoare (1969): partial correctness statements

$A\{G\}B =$ "if A holds before the execution of G then B holds afterwards" .

Recently Hoare was knighted by the British Queen.

Pratt (1976): logic of programs,

$[C]\phi = \phi$ holds while C is executed,

each $[C]$ is an **S4**-modality, Kripke style semantics where possible worlds are
machine states

Pnueli (1977): branching temporal logic = logic of concurrency, Turing
award in CS

*Logic of Knowledge: a core AI topic, $K_A(\phi)$ = "agent A knows ϕ ", multiple modalities **S4** and **S5**, negative introspection, common knowledge operator.*

Logical Omniscience: *unrealistic assumption that an agent knows all logical consequences of its knowledge.*

Major Problem (Joe Halpern, Rohit Parikh, and others):
build a logic of knowledge that distinguishes "hard" and "easy" facts

Intuitionism: constructive approach to mathematics

Brouwer (1900s):

"It does not make sense to think of truth or falsity of a mathematical statement independently of our knowledge concerning the statement. A statement is *true* if we have a proof of it, and *false* if we can show that the assumption that there is a proof for the statement leads to a contradiction."

Constructive semantics problem

BHK problem: find the intended provability semantics of intuitionistic logic satisfying *BHK* conditions:

- a proof of $A \wedge B$ consists of a proof of A and a proof of B ,
- a proof of $A \vee B$ is given by presenting either a proof of A or a proof of B ,
- a proof of $A \rightarrow B$ is a construction which, given a proof of A returns a proof of B ,
- absurdity \perp is a proposition which has no proof, $\neg A$ is $A \rightarrow \perp$.

Crucial for understanding connections between computations and derivations!

Major models for intuitionistic logic

1. Algebraic semantics (Birkhoff, 1935)
2. Topological semantics (Stone, 1937; Tarski, 1938)
3. Realizability semantics (Kleene, 1945)
4. Beth models (1956)
5. Dialectica Interpretation (Gödel, 1958)
6. Curry - Howard isomorphism (1958)
7. Medvedev's logic of problems (1962)
8. Kripke models (1965)
9. Kuznetsov-Muravitsky-Goldblatt provability interpretation (1976)
10. Categorical semantics (Goldblatt, 1979)

None solves the original **BHK**-problem!

Intuitionistic system = classical system + effective \forall and \exists .

Existential property of intuitionistic systems:

a constructive proof of $\forall x \exists y A(x, y)$ yields computable term (program) $f(x)$ such that $\forall x A(x, f(x))$ holds.

Corollary: *intuitionistic correctness proof =
program + correctness proof =
verified program*

Downside: produces correct but computationally not optimal programs

Explicit tradition: Functions vs. Quantifiers:

Skolem (1920), Herbrand (1930), Gödel (1930):

quantifiers \forall, \exists are ghosts of functions, a precursor to automated reasoning!:

logic	explicit logic
$\forall x \exists y A(x, y)$	$A(x, f(x))$
$\exists x A(x) \rightarrow \exists y B(y)$	$A(x) \rightarrow B(f(x))$

By its nature the closest to Computer Science. Addresses the right set of questions: whether $f(x)$ is computable, feasible, etc.

Downside: Too many Skolem functions, unification problems.

Shönfinkel (1924), Church (1929, 1930):

λ -calculus = universal functional language, foundational motivations.

Normal form = the result of a computation,

normalization process = computation

McCarthy (1960): λ -calculus implemented in *LISP*,

universal machine = *LISP*-compiler in *LISP*.

Functional programming languages.

Curry (1934), Howard (1969):

typed λ -calculus, implemented in ML

$t:F \sim$ term t has type $F \sim t$ is a proof of the proposition F

proofs \sim functional programs

assumptions \sim initial data

deduction \sim execution sequence

Girard (1971): second order λ -calculus, yields the consistency of the second order arithmetic, is implemented in the prover *Coq*

Martin-Löf (1982): type theory with intuitionistic logic, is implemented in Constable's *NuPRL* prover at Cornell, verification, programming via proofs, formally verified mathematics

Constructive quantifiers = computable Skolem functions

*Good news: intuitionistic correctness proof =
= program + correctness proof = verified program*

*Bad news: the above scheme is not efficient so far. Too long detour:
formal specs $S(x, y) \mapsto$ constructive proof of $\forall x \exists y S(x, y) \mapsto$
 \mapsto computable Skolem function $y = f(x)$*

*Compromise: reverse engineering of proofs, write a proof of $\forall x \exists y S(x, y)$ tar-
geting a specific algorithm $y = f(x)$*

Gödel's provability problem

Gödel, 1933: provability is a modality $\Box F \sim$ "there exists a proof of F "

$$\begin{aligned}\Box A &\rightarrow \Box B \\ \exists x(x \text{ is a proof of } A) &\rightarrow \exists y(y \text{ is a proof of } B) \\ "x \text{ is a proof of } A" &\rightarrow "f(x) \text{ is a proof of } B" \\ "x:A &\rightarrow f(x):B\end{aligned}$$

Curry-Howard Isomorphism of proofs and programs: typed λ -terms (programs) correspond to a simple case when the provability does not iterate

$$\begin{aligned}\Box A_1, \dots, \Box A_n &\Rightarrow \Box B \\ x_1:A_1, \dots, x_n:A_n &\Rightarrow t(x_1, \dots, x_n):B\end{aligned}$$

The general case - iterated provability Gödel (1933):

Provability Calculus, a.k.a. S4

1. *Classical axioms and rules*

2. $\Box(F \rightarrow G) \rightarrow (\Box F \rightarrow \Box G)$ *(implicit application)*

3. $\Box F \rightarrow F$ *(reflexivity)*

4. $\Box F \rightarrow \Box \Box F$ *(implicit proof checker)*

5. *Internalization rule:*

$$\frac{\vdash F}{\vdash \Box F}$$

*Reflects the basic intuition of Provability as a logic operator.
Derives all basic facts about Provability.*

Examples of derivations in **S4**.

\Box and \wedge commute:

$$\begin{array}{ll} A \rightarrow (B \rightarrow A \wedge B) & A \wedge B \rightarrow A \\ \Box(A \rightarrow (B \rightarrow A \wedge B)) & \Box(A \wedge B \rightarrow A) \\ \Box A \rightarrow \Box(B \rightarrow A \wedge B) & \Box(A \wedge B) \rightarrow \Box A \\ \Box A \rightarrow (\Box B \rightarrow \Box(A \wedge B)) & \Box(A \wedge B) \rightarrow \Box B \\ (\Box A \wedge \Box B) \rightarrow \Box(A \wedge B) & \Box(A \wedge B) \rightarrow (\Box A \wedge \Box B) \end{array}$$

\Box factors out through \vee :

$$\begin{array}{l} A \rightarrow A \vee B \\ \Box(A \rightarrow A \vee B) \\ \Box A \rightarrow \Box(A \vee B) \\ \Box B \rightarrow \Box(A \vee B) \\ (\Box A \vee \Box B) \rightarrow \Box(A \vee B) \end{array}$$

But not $\Box(A \vee B) \rightarrow (\Box A \vee \Box B)$!
Consider B to be $\neg A$

Gödel's embedding of **Int** into **S4**:

1. translate **Int**-formula F into a classical language \square :

$$tr(F) = \text{“box each subformula of } F\text{”},$$

2. test the translation in **S4**:

$$\mathbf{Int} \text{ proves } F \Leftrightarrow \mathbf{S4} \text{ proves } tr(F)$$

(Gödel (1933), McKinsey & Tarski (1948))

The mission has not been accomplished though, since

***S4** itself was left without an exact provability model*

$$\mathbf{Int} \hookrightarrow \mathbf{S4} \hookrightarrow ? \hookrightarrow \mathbf{REAL PROOFS}$$

FMU on Provability (Gödel, 1931):

$\text{Proof}_T(X, Y) \sim$ “ X is a proof of Y ”

$\text{Provable}_T(Y) = \exists X \text{Proof}_T(X, Y) \sim Y$ is provable”

“ T is consistent” = $\text{Consis } T = \neg \text{Provable}_T(\text{false})$

Reflection scheme: $\text{Provable}_T(\phi) \rightarrow \phi$

Consistency is a special case of reflection:

$$\neg \text{Provable}_T(\text{false}) = \text{Provable}_T(\text{false}) \rightarrow \text{false}$$

Incompleteness Theorem:

T does not prove $\text{Consis } T$

Reflection is not provable:

T does not prove $\text{Provable}_T(\varphi) \rightarrow \varphi$

Corollary Gödel (1933): **S4 modality** \neq *Provable*(\cdot)

Indeed, $\Box(\Box F \rightarrow F)$ is provable in **S4**:

$\Box F \rightarrow F$ - reflexivity axiom

$\Box(\Box F \rightarrow F)$ - by Internalization rule

However, under the interpretation of \Box as *Provable* this asserts that reflection is internally provable

$$\text{Provable}_T(\text{Provable}_T(F) \rightarrow F)$$

which is false by Gödel's Incompleteness Theorem.

Gödel's problem: *find an exact provability semantics of S4.*

This loophole remained open for about 60 years.

1938 Gödel's provability problem:

find a system of proof terms that corresponds to **S4**.

If successful yields

- *complete logical description of provability (Gödel's problem)*
- *formalization of constructive semantics for intuitionistic logic (**BHK**-problem)*
- *quantitative logic of knowledge (logical omniscience problem)*
- *much richer type systems for programming languages (referential types, coding computations in types)*
- *etc.*

Complete solution has been found recently.

Proof Polynomials

Basis for all invariant propositional operations on proofs

variables x, y, z, \dots *ranging over proofs*

constants a, b, c, \dots *proofs of instances of logical axioms*

“.” is **application**: *applies $s:(F \rightarrow G)$ to $t:F$ and returns $(s \cdot t):G$*

“!” is **proof checking**: *computes $!t$ a proof of $t:F$*

“+” is **union**: *takes union (concatenation) of two proofs*

Proof Polynomials

Basis for all invariant propositional operations on proofs

variables x, y, z, \dots *ranging over proofs*

constants a, b, c, \dots *proofs of instances of logical axioms*

“.” is **application**: *applies $s:(F \rightarrow G)$ to $t:F$ and returns $(s \cdot t):G$*

“!” is **proof checking**: *computes $!t$ a proof of $t:F$*

“+” is **union**: *takes union (concatenation) of two proofs*

Logic of Propositions and Proofs

LP = classical logic + additional atoms $p:F$,
(p is a proof polynomial and F is a formula)

A0. classical axioms and rules

A1. $t:(F \rightarrow G) \rightarrow (s:F \rightarrow (t \cdot s):G)$

(application)

A2. $t:F \rightarrow F$

(explicit reflexivity)

A3. $t:F \rightarrow !t:(t:F)$

(proof checker)

A4. $s:F \rightarrow (s \dagger t):F, \quad t:F \rightarrow (s \dagger t):F$

(union)

R1. $A \vdash c:A$, where $A \in A0-A4$, c is a proof constant.

(axiom necessitation)

A distant relative: typed combinatory logic **CL**.

Combinatory terms have dual meaning as typed terms and as derivations in a Hilbert style proof system. Constant combinators stand for proofs of axioms:

$$\mathbf{k}^{A,B} : (A \rightarrow (B \rightarrow A)), \quad \mathbf{s}^{A,B,C} : [(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))]$$

Variables in **CL** denote unspecified proofs, the operation of application “.” corresponds to the rule *modus ponens*

$$t:(F \rightarrow G) \rightarrow (s:F \rightarrow (t \cdot s):G)$$

The whole of **CL** corresponds to a fragment of **S4** consisting only of formulas of the sort

$$\Box A_1 \wedge \dots \wedge \Box A_n \rightarrow \Box B,$$

where A_1, \dots, A_n, B do not contain modalities.

Examples of derivations

Derivation in **S4**

$$\begin{aligned} & A \rightarrow (B \rightarrow A \wedge B) \\ \Box & (A \rightarrow (B \rightarrow A \wedge B)) \\ \Box & A \rightarrow \Box(B \rightarrow A \wedge B) \\ \Box & A \rightarrow (\Box B \rightarrow \Box(A \wedge B)) \\ (\Box & A \wedge \Box B) \rightarrow \Box(A \wedge B) \end{aligned}$$

Derivation in **LP**

Examples of derivations

Derivation in **S4**

$$A \rightarrow (B \rightarrow A \wedge B)$$

$$\Box(A \rightarrow (B \rightarrow A \wedge B))$$

$$\Box A \rightarrow \Box(B \rightarrow A \wedge B)$$

$$\Box A \rightarrow (\Box B \rightarrow \Box(A \wedge B))$$

$$(\Box A \wedge \Box B) \rightarrow \Box(A \wedge B)$$

Derivation in **LP**

$$A \rightarrow (B \rightarrow A \wedge B)$$

Examples of derivations

Derivation in **S4**

$$\begin{aligned} & A \rightarrow (B \rightarrow A \wedge B) \\ \Box & (A \rightarrow (B \rightarrow A \wedge B)) \\ \Box & A \rightarrow \Box(B \rightarrow A \wedge B) \\ \Box & A \rightarrow (\Box B \rightarrow \Box(A \wedge B)) \\ (\Box & A \wedge \Box B) \rightarrow \Box(A \wedge B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} & A \rightarrow (B \rightarrow A \wedge B) \\ c: & (A \rightarrow (B \rightarrow A \wedge B)) \end{aligned}$$

Examples of derivations

Derivation in **S4**

$$\begin{aligned} & A \rightarrow (B \rightarrow A \wedge B) \\ \Box(A \rightarrow (B \rightarrow A \wedge B)) \\ \Box A \rightarrow \Box(B \rightarrow A \wedge B) \\ \Box A \rightarrow (\Box B \rightarrow \Box(A \wedge B)) \\ (\Box A \wedge \Box B) \rightarrow \Box(A \wedge B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} & A \rightarrow (B \rightarrow A \wedge B) \\ c:(A \rightarrow (B \rightarrow A \wedge B)) \\ x:A \rightarrow (c \cdot x):(B \rightarrow A \wedge B) \end{aligned}$$

Examples of derivations

Derivation in **S4**

$$\begin{aligned} & A \rightarrow (B \rightarrow A \wedge B) \\ \Box(A \rightarrow (B \rightarrow A \wedge B)) \\ \Box A \rightarrow \Box(B \rightarrow A \wedge B) \\ \Box A \rightarrow (\Box B \rightarrow \Box(A \wedge B)) \\ (\Box A \wedge \Box B) \rightarrow \Box(A \wedge B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} & A \rightarrow (B \rightarrow A \wedge B) \\ c:(A \rightarrow (B \rightarrow A \wedge B)) \\ x:A \rightarrow (c \cdot x):(B \rightarrow A \wedge B) \\ x:A \rightarrow (y:B \rightarrow ((c \cdot x) \cdot y):(A \wedge B)) \end{aligned}$$

Examples of derivations

Derivation in **S4**

$$\begin{aligned} & A \rightarrow (B \rightarrow A \wedge B) \\ \Box(A \rightarrow (B \rightarrow A \wedge B)) \\ \Box A \rightarrow \Box(B \rightarrow A \wedge B) \\ \Box A \rightarrow (\Box B \rightarrow \Box(A \wedge B)) \\ (\Box A \wedge \Box B) \rightarrow \Box(A \wedge B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} & A \rightarrow (B \rightarrow A \wedge B) \\ c:(A \rightarrow (B \rightarrow A \wedge B)) \\ x:A \rightarrow (c \cdot x):(B \rightarrow A \wedge B) \\ x:A \rightarrow (y:B \rightarrow ((c \cdot x) \cdot y):(A \wedge B)) \\ (x:A \wedge y:B) \rightarrow ((c \cdot x) \cdot y):(A \wedge B) \end{aligned}$$

This was an easy ride, straightforward from **S4**.

Examples of derivations.

Some problems on the way from **S4**:

Derivation in **S4**

$$A \rightarrow A \vee B$$

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

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

$$B \rightarrow A \vee B$$

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

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

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

Derivation in **LP**

Examples of derivations.

Some problems on the way from **S4**:

Derivation in **S4**

$$\begin{aligned} & A \rightarrow A \vee B \\ & \Box(A \rightarrow A \vee B) \\ & \Box A \rightarrow \Box(A \vee B) \\ & B \rightarrow A \vee B \\ & \Box(B \rightarrow A \vee B) \\ & \Box B \rightarrow \Box(A \vee B) \\ & (\Box A \vee \Box B) \rightarrow \Box(A \vee B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} & A \rightarrow A \vee B \\ & a:(A \rightarrow A \vee B) \\ & x:A \rightarrow (a \cdot x):(A \vee B) \end{aligned}$$

Examples of derivations.

Some problems on the way from **S4**:

Derivation in **S4**

$$\begin{aligned} &A \rightarrow A \vee B \\ &\Box(A \rightarrow A \vee B) \\ &\Box A \rightarrow \Box(A \vee B) \\ &B \rightarrow A \vee B \\ &\Box(B \rightarrow A \vee B) \\ &\Box B \rightarrow \Box(A \vee B) \\ &(\Box A \vee \Box B) \rightarrow \Box(A \vee B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} &A \rightarrow A \vee B \\ &a:(A \rightarrow A \vee B) \\ &x:A \rightarrow (a \cdot x):(A \vee B) \\ &B \rightarrow A \vee B \\ &b:(B \rightarrow A \vee B) \\ &y:B \rightarrow (b \cdot y):(A \vee B) \end{aligned}$$

Examples of derivations.

Some problems on the way from **S4**:

Derivation in **S4**

$$\begin{aligned} &A \rightarrow A \vee B \\ &\Box(A \rightarrow A \vee B) \\ &\Box A \rightarrow \Box(A \vee B) \\ &B \rightarrow A \vee B \\ &\Box(B \rightarrow A \vee B) \\ &\Box B \rightarrow \Box(A \vee B) \\ &(\Box A \vee \Box B) \rightarrow \Box(A \vee B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} &A \rightarrow A \vee B \\ &a:(A \rightarrow A \vee B) \\ &x:A \rightarrow (a \cdot x):(A \vee B) \\ &B \rightarrow A \vee B \\ &b:(B \rightarrow A \vee B) \\ &y:B \rightarrow (b \cdot y):(A \vee B) \end{aligned}$$

Orange parts are different, and we cannot just repeat the corresponding **S4** step. Operation \vdash is needed!

Examples of derivations.

Some problems on the way from **S4**:

Derivation in **S4**

$$\begin{aligned} &A \rightarrow A \vee B \\ &\Box(A \rightarrow A \vee B) \\ &\Box A \rightarrow \Box(A \vee B) \\ &B \rightarrow A \vee B \\ &\Box(B \rightarrow A \vee B) \\ &\Box B \rightarrow \Box(A \vee B) \\ &(\Box A \vee \Box B) \rightarrow \Box(A \vee B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} &A \rightarrow A \vee B \\ &a:(A \rightarrow A \vee B) \\ &x:A \rightarrow (a \cdot x):(A \vee B) \quad [\rightarrow (a \cdot x + b \cdot y):(A \vee B)] \\ &B \rightarrow A \vee B \\ &b:(B \rightarrow A \vee B) \\ &y:B \rightarrow (b \cdot y):(A \vee B) \quad [\rightarrow (a \cdot x + b \cdot y):(A \vee B)] \end{aligned}$$

Examples of derivations.

Some problems on the way from **S4**:

Derivation in **S4**

$$\begin{aligned} &A \rightarrow A \vee B \\ &\Box(A \rightarrow A \vee B) \\ &\Box A \rightarrow \Box(A \vee B) \\ &B \rightarrow A \vee B \\ &\Box(B \rightarrow A \vee B) \\ &\Box B \rightarrow \Box(A \vee B) \\ &(\Box A \vee \Box B) \rightarrow \Box(A \vee B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} &A \rightarrow A \vee B \\ &a:(A \rightarrow A \vee B) \\ &x:A \rightarrow (a \cdot x):(A \vee B) [\rightarrow (a \cdot x + b \cdot y):(A \vee B)] \\ &B \rightarrow A \vee B \\ &b:(B \rightarrow A \vee B) \\ &y:B \rightarrow (b \cdot y):(A \vee B) [\rightarrow (a \cdot x + b \cdot y):(A \vee B)] \\ &(x:A \vee y:B) \rightarrow (a \cdot x + b \cdot y):(A \vee B) \end{aligned}$$

Examples of derivations. All three operations are needed

Derivation in **S4**

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

Derivation in **LP**

Examples of derivations. All three operations are needed

Derivation in **S4**

$$\begin{aligned} &\Box A \rightarrow \Box A \vee \Box B \\ &\Box(\Box A \rightarrow \Box A \vee \Box B) \\ &\Box A \rightarrow \Box\Box A \\ &\Box\Box A \rightarrow \Box(\Box A \vee \Box B) \\ &\Box A \rightarrow \Box(\Box A \vee \Box B) \\ &\Box B \rightarrow \Box A \vee \Box B \\ &\Box(\Box B \rightarrow \Box A \vee \Box B) \\ &\Box B \rightarrow \Box\Box B \\ &\Box\Box B \rightarrow \Box(\Box A \vee \Box B) \\ &\Box B \rightarrow \Box(\Box A \vee \Box B) \\ &\Box A \vee \Box B \rightarrow \Box(\Box A \vee \Box B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} &x:A \rightarrow x:A \vee y:B \\ &a:(x:A \rightarrow x:A \vee y:B) \\ &x:A \rightarrow !x:x:A \\ &!x:x:A \rightarrow (a \cdot !x):(x:A \vee y:B) \\ &x:A \rightarrow (a \cdot !x):(x:A \vee y:B) \end{aligned}$$

Examples of derivations. All three operations are needed

Derivation in **S4**

$$\begin{aligned} &\Box A \rightarrow \Box A \vee \Box B \\ &\Box(\Box A \rightarrow \Box A \vee \Box B) \\ &\Box A \rightarrow \Box\Box A \\ &\Box\Box A \rightarrow \Box(\Box A \vee \Box B) \\ &\Box A \rightarrow \Box(\Box A \vee \Box B) \\ &\Box B \rightarrow \Box A \vee \Box B \\ &\Box(\Box B \rightarrow \Box A \vee \Box B) \\ &\Box B \rightarrow \Box\Box B \\ &\Box\Box B \rightarrow \Box(\Box A \vee \Box B) \\ &\Box B \rightarrow \Box(\Box A \vee \Box B) \\ &\Box A \vee \Box B \rightarrow \Box(\Box A \vee \Box B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} &x:A \rightarrow x:A \vee y:B \\ &a:(x:A \rightarrow x:A \vee y:B) \\ &x:A \rightarrow !x:x:A \\ &!x:x:A \rightarrow (a \cdot !x):(x:A \vee y:B) \\ &x:A \rightarrow (a \cdot !x):(x:A \vee y:B) \\ &y:B \rightarrow x:A \vee y:B \\ &b:(y:B \rightarrow x:A \vee y:B) \\ &y:B \rightarrow !y:y:B \\ &!y:y:B \rightarrow (b \cdot !y):(x:A \vee y:B) \\ &y:B \rightarrow (b \cdot !y):(x:A \vee y:B) \end{aligned}$$

Examples of derivations. All three operations are needed

Derivation in **S4**

$$\begin{aligned} &\Box A \rightarrow \Box A \vee \Box B \\ &\Box(\Box A \rightarrow \Box A \vee \Box B) \\ &\Box A \rightarrow \Box\Box A \\ &\Box\Box A \rightarrow \Box(\Box A \vee \Box B) \\ &\Box A \rightarrow \Box(\Box A \vee \Box B) \\ &\Box B \rightarrow \Box A \vee \Box B \\ &\Box(\Box B \rightarrow \Box A \vee \Box B) \\ &\Box B \rightarrow \Box\Box B \\ &\Box\Box B \rightarrow \Box(\Box A \vee \Box B) \\ &\Box B \rightarrow \Box(\Box A \vee \Box B) \\ &\Box A \vee \Box B \rightarrow \Box(\Box A \vee \Box B) \end{aligned}$$

Derivation in **LP**

$$\begin{aligned} &x:A \rightarrow x:A \vee y:B \\ &a:(x:A \rightarrow x:A \vee y:B) \\ &x:A \rightarrow !x:x:A \\ &!x:x:A \rightarrow (a \cdot !x):(x:A \vee y:B) \\ &x:A \rightarrow (a \cdot !x):(x:A \vee y:B) [\rightarrow (a \cdot !x + b \cdot !y):(\dots)] \\ &y:B \rightarrow x:A \vee y:B \\ &b:(y:B \rightarrow x:A \vee y:B) \\ &y:B \rightarrow !y:y:B \\ &!y:y:B \rightarrow (b \cdot !y):(x:A \vee y:B) \\ &y:B \rightarrow (b \cdot !y):(x:A \vee y:B) [\rightarrow (a \cdot !x + b \cdot !y):(\dots)] \\ &x:A \vee y:B \rightarrow (a \cdot !x + b \cdot !y):(x:A \vee y:B) \end{aligned}$$

Comparing formats

Type (logic) derivation
(plain types - propositions)

$$A \rightarrow B, A \vdash B$$

λ -derivation (Curry-Howard)

$$s:(A \rightarrow B), t:A \vdash (s \cdot t):B$$

(plain typed λ -terms, explicit, but no proof iterations allowed)

Modal derivation (in **S4**)

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

(provability iterates, but is implicit)

Proof polynomial derivation

$$x:A \vee y:B \vdash (a \cdot !x + b \cdot !y):(x:A \vee y:B)$$

(provability is explicit

$$a:(x:A \rightarrow x:A \vee y:B)$$

and iterates freely)

$$b:(y:B \rightarrow x:A \vee y:B)$$

On the expressive power of Proof Polynomials.

LP enjoys the constructive form of Internalization:

$$\frac{A_1, \dots, A_n \vdash B}{x_1:A_1, \dots, x_n:A_n \vdash t(x_1, \dots, x_n):B}$$

for all **LP** formulas A_1, A_2, \dots, A_n, B . The famous Curry-Howard isomorphism covers only a very simple special case of this rule when A_1, A_2, \dots, A_n, B are boolean formulas, i.e. do not contain modalities.

Polymorphism: multi-conclusion proofs

Operation “+” yields multiple types:

Imagine we have $s:A$ and $t:B$. Then both holds: $(s + t):A$ and $(s + t):B$, i.e. term $s + t$ has types A and B .

Suppose, we want to restrict explicit modal considerations to single-conclusion proofs only. Then we will have some weird identities like $x:\top \rightarrow \neg x:(\top \wedge \top)$. This one, for example has a forgetful projection $\Box\top \rightarrow \neg\Box(\top \wedge \top)$ which is provably false in any normal modal logic.

Realization Theorem (S.A, 1995): **S4** proves F iff there is an assignment r of proof polynomials to all \square 's in F such that the corresponding realization F^r is derivable in **LP**.

The part “if” is straightforward: given an **LP**-derivation replace proof polynomials by empty \square 's and get a derivation in **S4**. Part “only if” is not at all easy. Let us try the “naive” approach: induction on a given derivation in **S4**. Realization of **S4** axioms is trivial. Step: *modus ponens*

$$\frac{A \rightarrow B, \quad A}{B}$$

By I.H., the premises are realizable $(A \rightarrow B)^r$ and A^r . Since r clearly commutes with \rightarrow , we have $A^r \rightarrow B^r$ and A^r . Therefore,

$$\frac{A^r \rightarrow B^r, \quad A^r}{B^r}$$

What is wrong with this “proof”?

Yes, you are right. In

$$\frac{A^r \rightarrow B^r, \quad A^r}{B^r}$$

those r 's in $A^r \rightarrow B^r$ and in A^r depend on derivations in **S4** of $A \rightarrow B$ and of A respectively, and thus are *different*. In order to make this step one has to reconcile realizations of $A \rightarrow B$ and A . In any case, such a realization step cannot be “local” and should depend on the whole derivation tree.

*True realization algorithm uses so-called normal (i.e. cut-free) derivations in **S4**.*

Reflective types

Type derivation:

$$A \vdash \underbrace{B \rightarrow A \wedge B}_{\text{type}}$$

λ -derivation:

$$x:A \vdash \underbrace{\lambda y. \mathbf{p}(x, y)}_{\text{term}} : \underbrace{(B \rightarrow A \wedge B)}_{\text{type}}$$

Reflexive λ -derivation:

$$u:x:A \vdash \underbrace{\lambda v. \mathbf{p}^2(u, v)}_{\text{term}} : \underbrace{(y:B \rightarrow \mathbf{p}(x, y):(A \wedge B))}_{\text{type}}$$

Reflective λ -calculus

Unified system of types (propositions) and λ -terms (proofs, programs)

(In the diagram below " \longrightarrow " denotes internalization)

$$\lambda^{(0)} = \mathbf{Int} \longrightarrow \lambda^{(1)} = \lambda\text{-calculus} \longrightarrow \lambda^{(2)} \longrightarrow \lambda^{(3)} \longrightarrow \dots$$

$$\text{proof polynomials} = \lambda^\infty = \bigcup \lambda^{(i)}$$

Curry - Howard isomorphism is a simple special case $\lambda^{(0)} \longrightarrow \lambda^{(1)}$

Research program:

upgrade existing systems of typed λ -terms by tractable self-referential mechanism as above (typed theories, programming languages, provers, etc.)

Knowledge with justifications

Proof polynomials made modal logic explicit (**van Benthem Problem**)

Joe Halpern: *An agent knows the product of two very large prime integers. In what sense does the agent know those primes? Another version: in what sense does an agent know all the tautologies?*

Research program:

Knowledge representation systems on the basis of explicit modal logics

$$p:F \sim \text{"}p \text{ is a justification for } F\text{"}$$
$$n:F \sim \text{"there is a proof of } F \text{ of length } \leq n\text{"}$$

$k:F$ holds for large k 's but does not hold for "small" k 's

Stability of verification

The common architecture of verification systems (Davis-Schwartz,1979):

verified rule: \Box "premises" \Rightarrow \Box "conclusion"

trusted core + extensions by verified rules.

Stability property: *"extended system = original system"*

Gödel Incompleteness Theorem yields that stability is not internally verifiable.

Explicit verification mechanism: *build a computable function $f(x)$ such that*

x :"premises" \Rightarrow $f(x)$:"conclusion"

Theorem: *Explicit stability is verifiable*

Circumvents Gödel's Incompleteness, covers all known and intended cases,
the right language for stability issues

Research program: Explicit reflection mechanisms for verification

Reflection in automated deduction

Goal: to build a proof systems matching conciseness as the natural reasoning (verification, formalization of mathematics, programming by extracts, etc.)

Challenge: reflective reasoning

"Similar to the previous argument with A instead of B ..."

Reflection \sim interpreter of a language in itself

Gödel numbering is universal but notoriously inefficient

A promising approach: incorporating reflection axiomatically

Research program:

Build feasible reflection for major provers (NuPRL, HOL, Coq, etc.).

Verify the proof of the Second Gödel Incompleteness Theorem

(a weak version of a much smaller First Incompleteness Theorem was verified at Stanford in the early 1990s).

Computational Logic: Program

1. Intuitionistic logic: semantics, proof systems, models.
2. Natural deduction and Gentzen proof systems.
3. Normalization of proofs in modal logics.
4. Embedding of intuitionistic logic to modal logic.
5. Lambek Calculus and grammatic types, Linear Logic.
6. Combinatory logic and typed λ -calculus. Church-Rosser property and strong normalization. Curry-Howard isomorphism of typed λ -terms and intuitionistic proofs.
7. Proof polynomials and proof-carrying formulas. Logic of Proofs. Internalization property. Realization of modal logic by proof polynomials. Solution of Gödel's problem.
8. Reflective λ -calculus. Curry-Howard automorphism.
9. Logical omniscience problem. Explicit epistemic logic approach.

Reading

- 1*. D. van Dalen, *Logic and Structure*, Springer-Verlag, third-fourth edition.
- 2*. A. Troelstra & H. Schwichtenberg, *Basic Proof Theory*, Cambridge University Press, 1996.
3. J.-Y. Girard, Y. Lafont & P. Taylor, *Proofs and Types*, Cambridge University Press, Cambridge 1989.
4. S. Artemov, "Explicit provability and constructive semantics", *Bulletin of Symbolic Logic*, volume 7, No.1, pp. 1-36, 2001

* = worth of buying