直觉主义逻辑与程序设计语言

王迪

北京大学计算机学院

2023 年 3 月 28 日

数学是什么？

柏拉图的数学哲学

- 数学是客观的抽象存在。
- 人类可以认知、理解数学。

布劳威尔的数学哲学[1]

- 数学是在进行（主观的）构造。
- 数学关乎能够被构造的知识而非某些（不可知的）绝对真理。

[1] L. E. J. Brouwer. 1907. *Over de Grondslagen der Wiskunde*. PhD thesis. Universiteit van Amsterdam. English translation "On the Foundations of Mathematics" in *Collected Works. I: Philosophy and Foundations of Mathematics*, Arend Heyting (ed.), North-Holland, 1975.

排中律

公理

对任意命题 A,要么 A 为真,要么非 A 为真,即 $A \vee \neg A$ 为真。

有问题吗?

布劳威尔点了点头

- 要说明 $A \vee B$ 为真,我们必须有 A 为真的证明或 B 为真的证明。
- 对于一个命题 A,我们可能既不知道如何证明 A,也不知道如何证否 A。

排中律

考虑“G = 哥德巴赫猜想”：任意偶数可以拆分为两个质数之和。

以布劳威尔的观点来看

- 由于哥德巴赫猜想尚未解决，我们不能说明 $G \vee \neg G$ 为真。
- 我们也不能说 $G \vee \neg G$ 不真，因为在未来人们可能解决该猜想。
- 但是，我们可以说明 $G \vee \neg G$ 不假。

原则 (直觉主义)

命题不具有绝对真值：命题 A 的真值定义为 A 为真的证明。

直觉主义逻辑是构造性的，而排中律是非构造性的。

直觉主义命题逻辑

定义 (直觉主义真值[2])

- 一个 $A \wedge B$(“A 且 B”)的证明由一个 A 的证明和一个 B 的证明组成。
- $\top$(“永真”)有一个平凡的、不包含任何信息的证明。
- 一个 $A \vee B$(“A 或 B”)的证明由一个 A 的证明或一个 B 的证明组成。
- 一个 $A \supset B$(“A 蕴含 B”)的证明是一个能够将 A 的证明转化为 B 的证明的方法。
- 不存在 $\bot$(“永假”)的证明。
- 另外,$\neg A$(“非 A”)可定义为 $A \supset \bot$,其证明是一个能够将 A 的证明转化为永假的证明的方法。

直觉主义要求,上述的方法必须是可以实际操作的构造方式。

[2] A. S. Troelstra and D. van Dalen. 1988. *Constructivism in Mathematics: An Introduction*. Volume 1. North-Holland.

跟编程语言有什么关系？

直觉主义要求，构造证明的方法必须是可以实际操作的构造方式。

观察

构造性的证明描述了算法：为达成某一目标而定义的可机械执行的有限长度的步骤序列。

观察

- 一个 $A \supset B$ 的证明是一个能够将 A 的证明转化为 B 的证明的方法。
- 一个 $A \to B$ 的函数是一个能够将 A 类型的值转化为 B 类型的值的算法。
  - 例如绝对值函数 $abs : \mathsf{int} \to \mathsf{int}$ 描述了从整数转化为另一个整数的算法。

在报告中，我们将讨论这种“相似性”代表了一种很强的内在联系。

跟编程语言有什么关系？

编程语言是人们实现算法的工具。
编程语言的特点由可以编写的程序和可供使用的类型决定。

原则

我们的口号是：

- 证明即是程序。
- 命题即是类型。

设计一个（直觉主义的）逻辑系统 $\Longleftrightarrow$ 设计一个（带类型的）编程语言

提纲

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

自然演绎 Natural Deduction[3]

原则

- 逻辑连结词的定义为推理它们构成的命题何时为真的规则，这些规则称为引入(*Introduction*)规则。
- 以引入规则为基础，设计使用这些连结词构成的命题的规则，这些规则称为消除(*Elimination*)规则。

我们区分命题(Proposition)和判断(Judgment)，一个判断表示我们可能知道的某种知识的构造。比如，A 是一个命题，“A 为真”(记作 A *true*)是一个判断。

定义 (推理规则)

推理规则的一般形式为

$$\frac{J_1 \quad \cdots \quad J_n}{J}\ \mathit{name}$$

其中判断 J_1 到 J_n 被称为前提(premise)，判断 J 被称为结论(conclusion)。

[3] G. Gentzen. 1935. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39, 176–210. English translation in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131, North-Holland, 1969. DOI: 10.1007/BF01201353; D. Prawitz. 1965. *Natural Deduction*. Almqvist & Wiksell.

合取

定义 (合取的引入规则)

$$\frac{A\ \mathit{true} \qquad B\ \mathit{true}}{A \wedge B\ \mathit{true}}\ \wedge I$$

如果我们知道 $A \wedge B$ 为真，那么我们可以推理出什么呢？

定义 (合取的消除规则)

$$\frac{A \wedge B\ \mathit{true}}{A\ \mathit{true}}\ \wedge E_1 \qquad \frac{A \wedge B\ \mathit{true}}{B\ \mathit{true}}\ \wedge E_2$$

永真

定义 (永真的引入规则)

$$\frac{}{\top\ true}\ \top\mathrm{I}$$

如果我们知道 $\top$ 为真，那么我们可以推理出什么呢？
因为引入 $\top$ 时不需要任何额外信息，所以永真并没有消除规则。

观察

- 永真可以看作零个命题的合取。
- 两个命题的合取对应两条消除规则，故永真对应零条消除规则。

假言推理

观察

如何理解下面这个推理？

$$\dfrac{\dfrac{(A \wedge B) \wedge C\ \mathit{true}}{A \wedge B\ \mathit{true}}\wedge E_1}{B\ \mathit{true}}\wedge E_2$$

我们是在假设 $(A \wedge B) \wedge C\ \mathit{true}$ 成立的情况下证明了 $B\ \mathit{true}$ 成立。

定义 (假言推理)

假言推理一般具有如下形式：

$$\begin{array}{c} J_1 \quad \cdots \quad J_n \\ \vdots \\ J \end{array}$$

这里的判断 J_1 到 J_n 为未证明的假设，判断 J 是结论。

蕴含

定义 (蕴含的引入规则)

$$\dfrac{\dfrac{}{\mathrm{A}\ \mathit{true}}\,\mathfrak{u}\quad\vdots\quad\mathrm{B}\ \mathit{true}}{\mathrm{A}\supset\mathrm{B}\ \mathit{true}}\,\supset\mathrm{I}^{\mathfrak{u}}$$

上述规则表达了“假设 A *true* 成立，那么 B *true* 成立”。我们为该假设添加了一个取名为 $\mathfrak{u}$ 的引入规则。

定义 (蕴含的消除规则)

$$\dfrac{\mathrm{A}\supset\mathrm{B}\ \mathit{true}\qquad\mathrm{A}\ \mathit{true}}{\mathrm{B}\ \mathit{true}}\,\supset\mathrm{E}$$

$\mathrm{A}\supset\mathrm{B}$ *true* 表明我们可以把一个 A *true* 的证明转化为一个 B *true* 的证明。

例子

$A \supset (B \supset A)\ \mathit{true}$

$$\dfrac{\dfrac{\dfrac{}{A\ \mathit{true}}\,u}{B \supset A\ \mathit{true}}\,\supset I^{w}}{A \supset (B \supset A)\ \mathit{true}}\,\supset I^{u}$$

$(A \wedge B) \supset (B \wedge A)\ \mathit{true}$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge B\ \mathit{true}}\,u}{B\ \mathit{true}}\,\wedge E_2 \qquad \dfrac{\dfrac{}{A \wedge B\ \mathit{true}}\,u}{A\ \mathit{true}}\,\wedge E_1}{B \wedge A\ \mathit{true}}\,\wedge I}{(A \wedge B) \supset (B \wedge A)\ \mathit{true}}\,\supset I^{u}$$

析取

定义 (析取的引入规则)

$$\frac{A\ \mathit{true}}{A \vee B\ \mathit{true}}\ \vee I_1 \qquad\qquad \frac{B\ \mathit{true}}{A \vee B\ \mathit{true}}\ \vee I_2$$

如果知道 $A \vee B$ 为真，那么我们可以推理出什么呢？
我们可以进行分类讨论！

定义 (析取的消除规则)

$$\frac{A \vee B\ \mathit{true} \qquad \begin{array}{c}\overline{A\ \mathit{true}}\ u \\ \vdots \\ C\ \mathit{true}\end{array} \qquad \begin{array}{c}\overline{B\ \mathit{true}}\ w \\ \vdots \\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\ \vee E^{u,w}$$

我们为两种情况分别引入的假设取名为 u 和 w。

永假

由于永假不应该有证明，所以没有引入规则。

定义 (永假的消除规则)

$$\frac{\bot\ true}{\mathrm{C}\ true}\ \bot\mathrm{E}$$

观察

- 永假可以看作零个命题的析取。
- 两个命题的析取对应两条引入规则，故永假对应零条引入规则。

例子：排中律不假

我们可以把否定定义为 $\neg A \stackrel{\text{def}}{=} A \supset \bot$。

$\neg\neg(A \vee \neg A)\ \mathit{true}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(A \vee \neg A) \supset \bot\ \mathit{true}}\,u \qquad \dfrac{\dfrac{}{A\ \mathit{true}}\,w}{A \vee \neg A\ \mathit{true}}\vee I_1}{\bot\ \mathit{true}}\supset E}{A \supset \bot\ \mathit{true}}\supset I^w}{A \vee \neg A\ \mathit{true}}\vee I_2 \qquad \dfrac{}{(A \vee \neg A) \supset \bot\ \mathit{true}}\,u}{\bot\ \mathit{true}}\supset E}{\neg(A \vee \neg A) \supset \bot\ \mathit{true}}\supset I^u$$

和谐性 Harmony

原则

在一个逻辑系统中，我们希望消除规则既不会太强，也不会太弱。

- 不会太强：对一个连结词应用消除规则不会获得比引入该连结词更多的信息。
- 不会太弱：对一个连结词应用消除规则后可以在得出的信息上重新对它进行证明。

在报告中，我们重点考察“不会太强”。
对于每个连结词，我们需要对一个“引入后就消除”的证明进行化简，这个步骤被称为局部归约。

定义 (局部归约)

我们用如下记号表示从推理 $\mathcal{D}$ 到 $\mathcal{D}'$ 的局部归约，其中两个推理的结论都是 $\mathrm{A}\ true$，归约表示我们把一个连结词引入后立即消除的证明进行了化简：

$$\begin{array}{c}\mathcal{D}\\ \mathrm{A}\ true\end{array} \Longrightarrow_{\mathrm{R}} \begin{array}{c}\mathcal{D}'\\ \mathrm{A}\ true\end{array}$$

合取的局部归约

$\wedge I$ 接着 $\wedge E_1$

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A\ true\end{array}\qquad\begin{array}{c}\mathcal{E}\\ B\ true\end{array}}{A \wedge B\ true}\wedge I}{A\ true}\wedge E_1 \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{D}\\ A\ true\end{array}$$

$\wedge I$ 接着 $\wedge E_2$

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A\ true\end{array}\qquad\begin{array}{c}\mathcal{E}\\ B\ true\end{array}}{A \wedge B\ true}\wedge I}{B\ true}\wedge E_2 \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{E}\\ B\ true\end{array}$$

蕴含的局部归约

⊃I 接着 ⊃E

我们用 $[\mathcal{D}/u]\mathcal{E}$ 表示把 $\mathcal{E}$ 中所有用到名为 u 的假设的地方替换为 $\mathcal{D}$ 后得到的推理。

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\\ A\ \mathit{true}\end{array}}{\begin{array}{c}\mathcal{E}\\ B\ \mathit{true}\end{array}}\,u}{A \supset B\ \mathit{true}}\,\supset I^{u} \qquad \mathcal{D}\\ A\ \mathit{true}}{B\ \mathit{true}}\,\supset E \quad \Longrightarrow_{R} \quad \begin{array}{c}[\mathcal{D}/u]\mathcal{E}\\ B\ \mathit{true}\end{array}$$

析取的局部归约

$\vee I_1$ 接着 $\vee E$

$$\dfrac{\dfrac{\mathcal{D}}{A\ \mathit{true}}}{A \vee B\ \mathit{true}}\vee I_1 \qquad \dfrac{\overset{u}{\overline{A\ \mathit{true}}}}{\mathcal{E}\ C\ \mathit{true}} \qquad \dfrac{\overset{w}{\overline{B\ \mathit{true}}}}{\mathcal{F}\ C\ \mathit{true}}$$

$$\dfrac{A \vee B\ \mathit{true} \qquad C\ \mathit{true} \qquad C\ \mathit{true}}{C\ \mathit{true}}\vee E^{u,w} \quad \Longrightarrow_R \quad \begin{array}{c}[\mathcal{D}/u]\mathcal{E}\\ C\ \mathit{true}\end{array}$$

$\vee I_2$ 接着 $\vee E$

$$\dfrac{\dfrac{\mathcal{D}}{B\ \mathit{true}}}{A \vee B\ \mathit{true}}\vee I_2 \qquad \dfrac{\overset{u}{\overline{A\ \mathit{true}}}}{\mathcal{E}\ C\ \mathit{true}} \qquad \dfrac{\overset{w}{\overline{B\ \mathit{true}}}}{\mathcal{F}\ C\ \mathit{true}}$$

$$\dfrac{A \vee B\ \mathit{true} \qquad C\ \mathit{true} \qquad C\ \mathit{true}}{C\ \mathit{true}}\vee E^{u,w} \quad \Longrightarrow_R \quad \begin{array}{c}[\mathcal{D}/w]\mathcal{F}\\ C\ \mathit{true}\end{array}$$

提纲

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

柯里-霍华德同构 Curry-Howard Correspondence[4]

前面我们考虑了 A *true* 这种判断形式。
这里我们考虑一种新的判断形式 $M : A$，表示 M 是命题 A 的一个证明对象。

原则 (柯里-霍华德同构)

$M : A$ 可以解释为 M 是具有类型 A 的一个程序。

原则

- 如果 $M : A$ 成立，那么 A *true* 成立。
- 如果 A *true* 成立，那么存在一个证明对象 M 使得 $M : A$ 成立。
- 每个 $M : A$ 的推理对应一个结构完全相同的 A *true* 的推理。

在自然演绎的推理规则中标注相应的证明对象

[4] W. A. Howard. 1969. The Formulae-as-Types Notion of Construction. Unpublished note. An annotated version appeared in: *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 479–490, Academic Press (1980). (1969).

合取

定义 (合取的引入规则)

$$\frac{M : A \qquad N : B}{\langle M, N \rangle : A \wedge B} \wedge I$$

$\langle M, N \rangle$ 表示我们把两个证明对象 M 和 N 通过一个二元组结合起来。

定义 (合取的消除规则)

$$\frac{M : A \wedge B}{\textbf{fst}\ M : A} \wedge E_1 \qquad \frac{M : A \wedge B}{\textbf{snd}\ M : B} \wedge E_2$$

fst 和 **snd** 分别表示取出二元组的第一项和第二项。

观察

作为合取命题的 $A \wedge B$ 对应编程语言中的二元组类型 $A \times B$。

永真

定义 (永真的引入规则)

$$\frac{}{\langle\rangle : \top}\ \top\mathrm{I}$$

$\langle\rangle$ 是一个平凡的证明对象，它不携带任何信息。

观察

作为永真命题的 $\top$ 对应编程语言中的零元组类型，在报告中我们记为 one。

蕴含

$A \supset B$ *true* 的证明表达了一个把 A *true* 的证明转化为 B *true* 的证明的函数。

定义 (λ 表达式[5])

在数学中，我们常常把以 x 为自变量的函数 f 记为“$f(x) =$ 带 x 的表达式”。例如，$f(x) = x^3 + x + 1$。
在编程语言中，我们可以用 λ 表达式来显式构造一个函数：$f = \lambda x.\ x^3 + x + 1$。

定义 (蕴含的引入规则)

$$\dfrac{\dfrac{}{u : A}\,u \\ \vdots \\ M : B}{\lambda u.\ M : A \supset B}\,\supset I^u$$

可以看到，我们引入的假设命名 u 对应函数的自变量。

[5] A. Church and J. B. Rosser. 1936. Some Properties of Conversion. *Trans. of the American Math. Society*, 39, 3, 472–482. DOI: 10.2307/1989762.

蕴含

我们把函数调用记为 $M\,N$，其中 M 为函数，N 为参数。

定义 (蕴含的消除规则)

$$\frac{M : A \supset B \qquad N : A}{M\,N : B}\ \supset E$$

观察

作为蕴含命题的 $A \supset B$ 对应编程语言中的函数类型 $A \to B$。

析取

定义 (析取的引入规则)

$$\frac{M : A}{\mathbf{inl}\ M : A \vee B}\ \vee I_1 \qquad\qquad \frac{M : B}{\mathbf{inr}\ M : A \vee B}\ \vee I_2$$

我们用 **inl** 和 **inr** 显式表示是证明了哪种情况。

定义 (析取的消除规则)

$$\frac{M : A \vee B \qquad \begin{array}{c}\overline{u : A}\ u \\ \vdots \\ N : C\end{array} \qquad \begin{array}{c}\overline{w : B}\ w \\ \vdots \\ P : C\end{array}}{\mathbf{case}\ M\ (\mathbf{inl}\ u \Rightarrow N \mid \mathbf{inr}\ w \Rightarrow P) : C}\ \vee E^{u,w}$$

我们用 **case** 来组织分类讨论的证明对象。

析取

观察

作为析取命题的 $A \vee B$ 对应编程语言中的枚举类型 $A + B$。

用枚举类型来实现布尔类型

我们可以定义 $\mathsf{bool} \stackrel{\text{def}}{=} \mathsf{one} + \mathsf{one}$。

那么我们可以定义 $\mathbf{true} \stackrel{\text{def}}{=} \mathbf{inl}\ \langle\rangle$，$\mathbf{false} \stackrel{\text{def}}{=} \mathbf{inr}\ \langle\rangle$。

进一步我们可以定义分支语句：$\mathbf{if}\ M\ \mathbf{then}\ N\ \mathbf{else}\ P \stackrel{\text{def}}{=} \mathbf{case}\ M\ (\mathbf{inl}\ \_ \Rightarrow N \mid \mathbf{inr}\ \_ \Rightarrow P)$。

永假

定义 (永假的消除规则)

$$\frac{M : \bot}{\textbf{abort}\ M : C}\ \bot E$$

永假不存在证明，那么如果 M 是 $\bot$ 的证明，那说明发生了错误，于是我们可以用 **abort** 来表示。

观察

作为永假命题的 $\bot$ 对应编程语言中的空枚举类型，在报告中我们记为 zero。

例子：排中律不假

$\neg\neg(A \vee \neg A)$

$$\dfrac{\dfrac{\dfrac{}{u : (A \vee \neg A) \supset \bot}\,u \qquad \dfrac{\dfrac{}{w : A}\,w}{\mathbf{inl}\ w : A \vee \neg A}\vee I_1}{u\,(\mathbf{inl}\ w) : \bot}\supset E}{\lambda w.\ u\,(\mathbf{inl}\ w) : A \supset \bot}\supset I^{w} \qquad \dfrac{\dfrac{\dfrac{}{u : (A \vee \neg A) \supset \bot}\,u \qquad \dfrac{\lambda w.\ u\,(\mathbf{inl}\ w) : A \supset \bot}{\mathbf{inr}\ (\lambda w.\ u\,(\mathbf{inl}\ w)) : A \vee \neg A}\vee I_2}{u\,(\mathbf{inr}\ (\lambda w.\ u\,(\mathbf{inl}\ w))) : \bot}\supset E}{\lambda u.\ u\,(\mathbf{inr}\ (\lambda w.\ u\,(\mathbf{inl}\ w))) : \neg(A \vee \neg A) \supset \bot}\supset I^{u}$$

小结：证明即是程序，命题即是类型

命题	规则	证明对象	类型
$A \wedge B$	$\wedge I$	$\langle \_, \_ \rangle$	$A \times B$
	$\wedge E_1, \wedge E_2$	**fst** _, **snd** _	
$A \supset B$	$\supset I^x$	$\lambda x.\ \_$	$A \to B$
	$\supset E$	$\_\ \_$	
$A \vee B$	$\vee I_1, \vee I_2$	**inl** _, **inr** _	$A + B$
	$\vee E^{u,w}$	**case** _ (**inl** $u \Rightarrow$ _ | **inr** $w \Rightarrow$ _)	
$\top$	$\top I$	$\langle\rangle$	one
$\bot$	$\bot E$	**abort** _	zero

不禁要问

程序是可以运行的，既然证明即是程序，那么证明要怎么运行呢？

证明归约即是计算

原则

通过在证明上进行归约来进行计算，以此实现“证明的运行”。

回顾

在推理规则和谐性的讨论中，我们引入了局部归约。

定义 (局部归约)

我们用如下记号表示从推理 $\mathcal{D}$ 到 $\mathcal{D}'$ 的局部归约，其中两个推理的结论都是 $\mathsf{A}\ true$，归约表示我们把一个连结词引入后立即消除的证明进行了化简：

$$\begin{array}{c}\mathcal{D}\\ \mathsf{A}\ true\end{array} \Longrightarrow_{\mathsf{R}} \begin{array}{c}\mathcal{D}'\\ \mathsf{A}\ true\end{array}$$

证明归约：合取

定义 (合取的局部归约)

$$\dfrac{\dfrac{\begin{array}{cc}\mathcal{D} & \mathcal{E} \\ A\ \mathit{true} & B\ \mathit{true}\end{array}}{A \wedge B\ \mathit{true}}\wedge I}{A\ \mathit{true}}\wedge E_1 \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{D} \\ A\ \mathit{true}\end{array}$$

$$\dfrac{\dfrac{\begin{array}{cc}\mathcal{D} & \mathcal{E} \\ A\ \mathit{true} & B\ \mathit{true}\end{array}}{A \wedge B\ \mathit{true}}\wedge I}{B\ \mathit{true}}\wedge E_2 \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{E} \\ B\ \mathit{true}\end{array}$$

定义 (合取的证明归约)

$$\dfrac{\dfrac{\begin{array}{cc}\mathcal{D} & \mathcal{E} \\ M : A & N : B\end{array}}{\langle M, N\rangle : A \wedge B}\wedge I}{\mathbf{fst}\ \langle M, N\rangle : A}\wedge E_1 \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{D} \\ M : A\end{array}$$

$$\dfrac{\dfrac{\begin{array}{cc}\mathcal{D} & \mathcal{E} \\ M : A & N : B\end{array}}{\langle M, N\rangle : A \wedge B}\wedge I}{\mathbf{snd}\ \langle M, N\rangle : B}\wedge E_1 \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{E} \\ N : B\end{array}$$

即 $\mathbf{fst}\ \langle M, N\rangle \Longrightarrow_R M$ 和 $\mathbf{snd}\ \langle M, N\rangle \Longrightarrow_R N$。

证明归约：蕴含

函数调用就是把函数的自变量替换为实际的参数。

例子

$(\lambda x.\ x^3 + x + 1)\ 2$ 这一函数调用需要把函数定义中的 x 替换为 2，可得 $2^3 + 2 + 1$，并可进一步计算出 11。

一般地，我们用 $[N/x]M$ 表示把 M 中的 x 都替换为 N 得到的结果[6]。

例子

$[2/x](x^3 + x + 1)$ 得到 $2^3 + 2 + 1$。

[6] 在进行替换时需要小心注意 N 中用到的变量不能在 M 中被绑定。当然，我们总是能通过对变量重命名来规避这个问题。

证明归约：蕴含

定义 (蕴含的局部归约)

$$\dfrac{\dfrac{\dfrac{}{A\ \mathit{true}}\,u}{\mathcal{E}}}{\dfrac{B\ \mathit{true}}{A \supset B\ \mathit{true}}\,\supset I^u} \qquad \dfrac{\mathcal{D}}{A\ \mathit{true}} \qquad \supset E \qquad \Longrightarrow_R \qquad \begin{array}{c}[\mathcal{D}/u]\mathcal{E}\\ B\ \mathit{true}\end{array}$$

定义 (蕴含的证明归约)

$$\dfrac{\dfrac{\dfrac{}{u : A}\,u}{\mathcal{E}}}{\dfrac{M : B}{\lambda u.\, M : A \supset B}\,\supset I^u} \qquad \dfrac{\mathcal{D}}{N : A} \qquad \supset E \qquad \Longrightarrow_R \qquad \begin{array}{c}[\mathcal{D}/u]\mathcal{E}\\ [M/u]N : B\end{array}$$

即 $(\lambda x.\, M)\, N \Longrightarrow_R [N/x]M$。

证明归约：析取

定义 (析取的局部归约之一)

$$\dfrac{\dfrac{\mathcal{D}}{A\ \mathit{true}}}{\dfrac{A\ \mathit{true}}{A \vee B\ \mathit{true}}\vee I_1} \quad \dfrac{}{A\ \mathit{true}}\,u \quad \dfrac{\mathcal{E}}{C\ \mathit{true}} \quad \dfrac{}{B\ \mathit{true}}\,w \quad \dfrac{\mathcal{F}}{C\ \mathit{true}}}{C\ \mathit{true}}\vee E^{u,w} \quad \Longrightarrow_R \quad \dfrac{[\mathcal{D}/u]\mathcal{E}}{C\ \mathit{true}}$$

定义 (析取的证明归约之一)

$$\dfrac{\dfrac{\mathcal{D}}{M : A}}{\dfrac{M : A}{\mathbf{inl}\ M : A \vee B}\vee I_1} \quad \dfrac{}{u : A}\,u \quad \dfrac{\mathcal{E}}{N : C} \quad \dfrac{}{w : B}\,w \quad \dfrac{\mathcal{F}}{P : C}}{\mathbf{case}\ (\mathbf{inl}\ M)\ (\mathbf{inl}\ u \Rightarrow N \mid \mathbf{inr}\ w \Rightarrow P) : C}\vee E^{u,w} \quad \Longrightarrow_R \quad \dfrac{[\mathcal{D}/u]\mathcal{E}}{[M/u]N : C}$$

即 $\mathbf{case}\ (\mathbf{inl}\ M)\ (\mathbf{inl}\ u \Rightarrow N \mid \mathbf{inr}\ w \Rightarrow P) \Longrightarrow_R [M/u]N$。
类似地，有 $\mathbf{case}\ (\mathbf{inr}\ M)\ (\mathbf{inl}\ u \Rightarrow N \mid \mathbf{inr}\ w \Rightarrow P) \Longrightarrow_R [M/w]P$。

小结：证明归约即是计算

类型	证明归约		
$A \times B$	$\mathbf{fst}\ \langle M, N\rangle$	$\Longrightarrow_R$	M
	$\mathbf{snd}\ \langle M, N\rangle$	$\Longrightarrow_R$	N
$A \to B$	$(\lambda u.\ M)\ N$	$\Longrightarrow_R$	$[N/u]M$
$A + B$	$\mathbf{case}\ (\mathbf{inl}\ M)\ (\mathbf{inl}\ u \Rightarrow N \mid \mathbf{inr}\ w \Rightarrow P)$	$\Longrightarrow_R$	$[M/u]N$
	$\mathbf{case}\ (\mathbf{inr}\ M)\ (\mathbf{inl}\ u \Rightarrow N \mid \mathbf{inr}\ w \Rightarrow P)$	$\Longrightarrow_R$	$[M/w]P$
one	对于 $\langle\rangle$ 无归约规则		
zero	对于 $\mathbf{abort}\ M$ 无归约规则		

提纲

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

函数式编程

观察

直觉主义命题逻辑与(简单类型的)函数式编程间存在柯里-霍华德同构。

类型	程序语法	证明对象	程序语法
$A \times B$	`'a * 'b`	$\langle M, N \rangle$	`(M, N)`
		$\mathbf{fst}\ M, \mathbf{snd}\ M$	`fst M, snd M`
$A \to B$	`'a -> 'b`	$\lambda x.\ M$	`fun x -> M`
		$M\ N$	`M(N)`
$A + B$	`('a,'b) sum`	$\mathbf{inl}\ M, \mathbf{inr}\ M$	`Inl(M), Inr(M)`
		$\mathbf{case}\ M\ (\mathbf{inl}\ u \Rightarrow N \mid \mathbf{inr}\ w \Rightarrow P)$	`match M with Inl(u) -> N | Inr(w) -> P`
one	`unit`	$\langle\rangle$	`()`
zero	`zero`	$\mathbf{abort}\ M$	`match M with _ -> .`

用函数式编程做证明

类型定义

```
(* internal: 'a * 'b *)
(* internal: 'a -> 'b *)
type ('a,'b) sum = Inl of 'a | Inr of 'b
(* internal: unit *)
type zero = |
```

$A \supset (B \supset A)$

```
let ex1 : 'a -> ('b -> 'a) =
  fun x -> fun y -> x
```

$(A \wedge B) \supset (B \wedge A)$

```
let ex2 : ('a * 'b) -> ('b * 'a) =
  fun x -> (snd x, fst x)
```

用函数式编程做证明

$(A \vee B) \supset (B \vee A)$

```
let ex3 : ('a,'b) sum -> ('b,'a) sum =
  fun x -> match x with Inl(y) -> Inr(y) | Inr(z) -> Inl(z)
```

$(A \wedge (A \supset B)) \supset B$

```
let ex4 : 'a * ('a -> 'b) -> 'b =
  fun x -> (snd x)(fst x)
```

$\neg(A \wedge \neg A)$

```
let ex5 : ('a * ('a -> zero)) -> zero =
  fun x -> (snd x)(fst x)
```

$\neg\neg(A \vee \neg A)$

```
let ex6 : (('a,('a -> zero)) sum -> zero) -> zero =
  fun x -> x(Inr(fun y -> x(Inl(y))))
```

类型系统与假言推理

观察

函数式编程语言的类型系统通常通过形如 $\Gamma \vdash M : A$ 的判断的推理系统给出。
其中 Γ 为一组变量和类型的绑定：$x_1 : A_1, x_1 : A_2, \cdots, x_n : A_n$。

我们可以把自然演绎的推理系统重写为类似类型系统的模式。

定义 (假言推理)

假言推理一般具有如下形式：

$$\begin{array}{c} J_1 \quad \cdots \quad J_n \\ \vdots \\ J \end{array}$$

这里的判断 J_1 到 J_n 为未证明的假设，判断 J 是结论。

我们可以把上述假言推理建立的假言判断（Hypothetical Judgment）记为 $J_1, \cdots, J_n \vdash J$，并称 J_1 到 J_n 为前件（antecedent），称 J 为后件（succedent）。

合取

定义 (合取的引入与消除)

$$\frac{A\ \mathit{true} \qquad B\ \mathit{true}}{A \wedge B\ \mathit{true}}\ \wedge I \qquad \frac{A \wedge B\ \mathit{true}}{A\ \mathit{true}}\ \wedge E_1 \qquad \frac{A \wedge B\ \mathit{true}}{B\ \mathit{true}}\ \wedge E_2$$

定义 (使用假言判断的合取的引入与消除)

我们显式地把需要的假设作为前件记录在 Γ 中。

$$\frac{\Gamma \vdash A\ \mathit{true} \qquad \Gamma \vdash B\ \mathit{true}}{\Gamma \vdash A \wedge B\ \mathit{true}}\ \wedge I \qquad \frac{\Gamma \vdash A \wedge B\ \mathit{true}}{\Gamma \vdash A\ \mathit{true}}\ \wedge E_1 \qquad \frac{\Gamma \vdash A \wedge B\ \mathit{true}}{\Gamma \vdash B\ \mathit{true}}\ \wedge E_2$$

假言判断的自然演绎

$$\frac{}{\Gamma, A\ \mathit{true} \vdash A\ \mathit{true}}\ \mathrm{ID}$$

$$\frac{\Gamma \vdash A\ \mathit{true} \qquad \Gamma \vdash B\ \mathit{true}}{\Gamma \vdash A \wedge B\ \mathit{true}}\ \wedge\mathrm{I} \qquad \frac{\Gamma \vdash A \wedge B\ \mathit{true}}{\Gamma \vdash A\ \mathit{true}}\ \wedge\mathrm{E}_1 \qquad \frac{\Gamma \vdash A \wedge B\ \mathit{true}}{\Gamma \vdash B\ \mathit{true}}\ \wedge\mathrm{E}_2 \qquad \frac{}{\Gamma \vdash \top\ \mathit{true}}\ \top\mathrm{I}$$

$$\frac{\Gamma, A\ \mathit{true} \vdash B\ \mathit{true}}{\Gamma \vdash A \supset B\ \mathit{true}}\ \supset\mathrm{I} \qquad \frac{\Gamma \vdash A \supset B\ \mathit{true} \qquad \Gamma \vdash A\ \mathit{true}}{\Gamma \vdash B\ \mathit{true}}\ \supset\mathrm{E} \qquad \frac{\Gamma \vdash A\ \mathit{true}}{\Gamma \vdash A \vee B\ \mathit{true}}\ \vee\mathrm{I}_1$$

$$\frac{\Gamma \vdash B\ \mathit{true}}{\Gamma \vdash A \vee B\ \mathit{true}}\ \vee\mathrm{I}_2 \qquad \frac{\Gamma \vdash A \vee B\ \mathit{true} \qquad \Gamma, A\ \mathit{true} \vdash C\ \mathit{true} \qquad \Gamma, B\ \mathit{true} \vdash C\ \mathit{true}}{\Gamma \vdash C\ \mathit{true}}\ \vee\mathrm{E}$$

$$\frac{\Gamma \vdash \bot\ \mathit{true}}{\Gamma \vdash C\ \mathit{true}}\ \bot\mathrm{E}$$

类型系统(或,标注证明对象的假言判断的自然演绎)

$$\frac{}{\Gamma, u : A \vdash u : A}\ \mathrm{ID}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N \rangle : A \wedge B}\ \wedge\mathrm{I} \qquad \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \mathbf{fst}\ M : A}\ \wedge\mathrm{E}_1 \qquad \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \mathbf{snd}\ M : B}\ \wedge\mathrm{E}_2 \qquad \frac{}{\Gamma \vdash \langle\rangle : \top}\ \top\mathrm{I}$$

$$\frac{\Gamma, u : A \vdash M : B}{\Gamma \vdash \lambda u.\ M : A \supset B}\ \supset\mathrm{I} \qquad \frac{\Gamma \vdash M : A \supset B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B}\ \supset\mathrm{E} \qquad \frac{\Gamma \vdash M : A}{\Gamma \vdash \mathbf{inl}\ M : A \vee B}\ \vee\mathrm{I}_1$$

$$\frac{\Gamma \vdash M : B}{\Gamma \vdash \mathbf{inr}\ M : A \vee B}\ \vee\mathrm{I}_2 \qquad \frac{\Gamma \vdash M : A \vee B \qquad \Gamma, u : A \vdash N : C \qquad \Gamma, w : B \vdash P : C}{\Gamma \vdash \mathbf{case}\ M\ (\mathbf{inl}\ u \Rightarrow N \mid \mathbf{inr}\ w \Rightarrow P) : C}\ \vee\mathrm{E}$$

$$\frac{\Gamma \vdash M : \bot}{\Gamma \vdash \mathbf{abort}\ M : C}\ \bot\mathrm{E}$$

柯里-霍华德同构

逻辑系统		编程语言
命题	$\Longleftrightarrow$	类型
证明	$\Longleftrightarrow$	程序
证明的归约	$\Longleftrightarrow$	程序的计算

原则 (协同设计)

- 在设计编程语言时,同时研究其对应的逻辑系统。
- 这种协同设计有很多变量:判断的形式,推理的方法,归约的方式,相等的处理,...
- 虽然已有一些理论指导(如和谐性),但通常还是很有难度!

提纲

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

阶段化计算 Staged Computation

观察

在编程时，我们有时候希望在运行过程中，得到一个函数的特化版本(往往效率更高)。

例子

考虑幂函数 $pow : \text{int} \to \text{int} \to \text{int}$，第一个参数是指数(假定为非负整数)，第二个参数是底数。
那么在程序中，一般而言，我们要传入两个参数 n 和 b 后，才可以进行幂的运算：$(pow\ n)\ b$。

如果有一个代码类型 $\text{code}(A)$，其值表示一个类型为 A 的源程序，其作用为运行时代码生成：

例子

考虑幂函数 $spow : \text{int} \to \text{code}(\text{int} \to \text{int})$。
这个类型显式说明，在传入一个参数 n 后，$spow\ n$ 会生成一个以 b 为参数，计算 b^n 的函数的代码。

阶段化计算的类型系统

原则

我们用 **box** M 来表示把 M 的代码打包作为具有代码类型 $\mathsf{code}(A)$ 的对象(假设 M 的类型为 A)。
那么代码中不应该有任何未被绑定的变量,所以其类型规则应该为:

$$\frac{\cdot \vdash M : A}{\Gamma \vdash \mathbf{box}\, M : \mathsf{code}(A)}$$

原则

但是,如果 M 中用到的变量表示的是另一个良定义的代码对象,我们应该可以支持这种使用方式。

定义 (阶段化计算的假言判断)

$$\underbrace{u_1 : B_1, \cdots, u_k : B_k}_{\text{代码对象变量}} ; \underbrace{x_1 : A_1, \cdots, x_n : A_n}_{\text{普通值的变量}} \vdash M : A$$

阶段化计算的类型系统

定义 (代码类型的引入和消除规则)

$$\frac{\Omega;\cdot \vdash M : A}{\Omega;\Gamma \vdash \mathbf{box}\, M : \mathsf{code}(A)}\ \text{code-I} \qquad \frac{\Omega;\Gamma \vdash M : \mathsf{code}(A) \qquad \Omega, u : A;\Gamma \vdash N : C}{\Omega;\Gamma \vdash \mathbf{let\ box}\, u = M\, \mathbf{in}\, N : C}\ \text{code-E}$$

定义 (代码类型的局部归约)

$$\frac{\dfrac{\mathcal{D}}{\Omega;\cdot \vdash M : A}}{\Omega;\Gamma \vdash \mathbf{box}\, M : \mathsf{code}(A)}\ \text{code-I} \qquad \frac{\mathcal{E}}{\Omega, u : A;\Gamma \vdash N : C}}{\Omega;\Gamma \vdash \mathbf{let\ box}\, u = \mathbf{box}\, M\, \mathbf{in}\, N : C}\ \text{code-E} \Longrightarrow_R \begin{array}{c}[\mathcal{D}/u]\mathcal{E} \\ [M/u]N : C\end{array}$$

即 $\mathbf{let\ box}\, u = \mathbf{box}\, M\, \mathbf{in}\, N \Longrightarrow_R [M/u]N$。

阶段化计算的柯里-霍华德同构是什么[7]?

观察

- 代码对象中不应该有任何绑定到普通值的变量。
- 代码对象中可以用到表示其它的良定义代码对象的变量。

原则

- 证明对象中不应该使用任何假设的命题。
- 证明对象中可以用假设的方式使用无需任何假设即可证明的命题。

[7] R. Davies and F. Pfenning. 2001. A Modal Analysis of Staged Computation. *J. ACM*, 48, 555–604, 3. DOI: 10.1145/382780.382785.

一种模态逻辑

定义

我们说一个命题 A 是合理的（记作 A *valid*），如果 A *true* 有一个不需要任何假设的证明。

观察

引入对“合理”的判断后，我们可以试着在逻辑系统里回答：
如果 $A \supset B$ 和 A 都合理，那么 B 合不合理？如果 A 合理，那么“A 合理”这个命题合不合理？

定义

为了在逻辑系统里回答上述问题，我们引入一个命题 $\Box A$ 来内在化 A *valid*。
上面讨论的问题就变成考虑 $(\Box(A \supset B) \wedge \Box A) \supset \Box B$、$\Box A \supset \Box\Box A$ 是否为真。

观察

代码类型 code(A) 对应逻辑命题 $\Box A$！

直觉主义版本的模态逻辑 S4

定义 (模态逻辑的假言判断)

$$B_1\ \mathit{valid}, \cdots, B_k\ \mathit{valid}; A_1\ \mathit{true}, \cdots, A_n\ \mathit{true} \vdash A\ \mathit{true}$$

定义 (□ 的引入和消除规则)

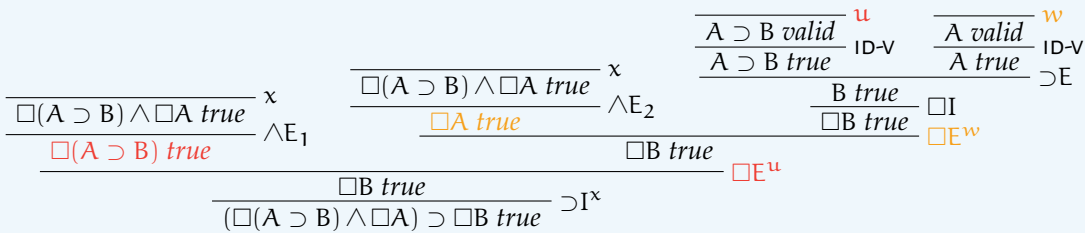
$$\frac{\Omega; \cdot \vdash A\ \mathit{true}}{\Omega; \Gamma \vdash \Box A\ \mathit{true}}\ \Box\mathrm{I} \qquad \frac{\Omega; \Gamma \vdash \Box A\ \mathit{true} \qquad \Omega, A\ \mathit{valid}; \Gamma \vdash C\ \mathit{true}}{\Omega; \Gamma \vdash C\ \mathit{true}}\ \Box\mathrm{E}$$

定义 (若 $A\ \mathit{valid}$ 成立，则 $A\ \mathit{true}$ 成立)

$$\frac{}{\Omega, A\ \mathit{valid}; \Gamma \vdash A\ \mathit{true}}\ \mathrm{ID\text{-}V}$$

例子

$(\Box(A \supset B) \wedge \Box A) \supset \Box B\ \mathit{true}$[8]



[8] 为了节省空间，在写证明时没有使用假言判断。

提纲

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

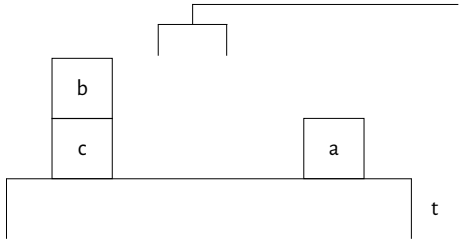
改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

在逻辑中表达状态变化

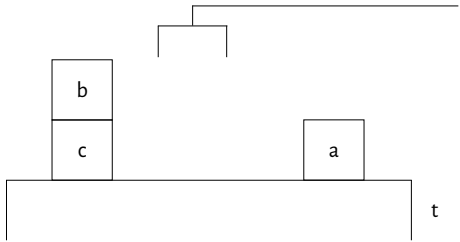


砖块世界

在桌子(t)上有若干砖块(a, b, c, $\cdots$)。我们用机械臂来移动这些砖块。机械臂只能抓取一个砖块。
如何在逻辑中表达上图中的状态?

- 谓词 on(x, y):砖块 x 在 y 上面;clear(x):砖块 x 上面没有别的砖块。
- 谓词 empty:机械臂闲置;holds(x):机械臂抓着砖块 x。

在逻辑中表达状态变化



$\text{empty}, \text{on}(a, t), \text{clear}(a), \text{on}(c, t), \text{on}(b, c), \text{clear}(b)$

如何在逻辑中表达...

若机械臂为空，砖块 x 在 y 上面且 x 上面为空，则可以状态转移到机械臂抓着砖块 x，砖块 y 上面为空。

$$\forall x.\ \forall y.\ (\text{empty} \wedge \text{on}(x, y) \wedge \text{clear}(x)) \supset (\text{holds}(x) \wedge \text{clear}(y))?$$

在逻辑中表达状态变化

如何在逻辑中表达...

若机械臂为空，砖块 x 在 y 上面且 x 上面为空，则可以状态转移到机械臂抓着砖块 x，砖块 y 上面为空。

$$\forall x.\ \forall y.\ (\mathsf{empty} \wedge \mathsf{on}(x, y) \wedge \mathsf{clear}(x)) \supset (\mathsf{holds}(x) \wedge \mathsf{clear}(y))?$$

观察

按照命题逻辑，上述蕴含命题的结论可以加强为 $\mathsf{empty} \wedge \mathsf{holds}(x) \wedge \mathsf{clear}(y)$，是一个矛盾的状态！

观察

在命题逻辑中，一旦我们证明了（或者通过假设引入了）A *true*，那么 A *true* 在当前证明中是一直存在的。

回顾

定义 (假言推理)

假言推理一般具有如下形式：

$$\begin{array}{ccc} \mathcal{J}_1 & \cdots & \mathcal{J}_n \\ & \vdots & \\ & \mathcal{J} & \end{array}$$

这里的判断 $\mathcal{J}_1$ 到 $\mathcal{J}_n$ 为未证明的假设，判断 $\mathcal{J}$ 是结论。

定义 (假言判断)

我们可以把上述假言推理建立的假言判断记为 $\mathcal{J}_1, \cdots, \mathcal{J}_n \vdash \mathcal{J}$，并称 $\mathcal{J}_1$ 到 $\mathcal{J}_n$ 为前件，称 $\mathcal{J}$ 为后件。

观察

在假言判断 $\mathcal{J}_1, \cdots, \mathcal{J}_n \vdash \mathcal{J}$ 的证明中，每个前件可以被使用任意多次。

线性逻辑 Linear Logic[9]

原则 (前件为资源，后件为目标)

在线性假言判断 $J_1, \cdots, J_n \Vdash J$ 的证明中，每个前件被使用且仅被使用一次。

$$\frac{}{A\ true \Vdash A\ true}\ \mathsf{ID}$$

定义 (同时合取)

我们用 $A \otimes B$ 表示可以使用一些资源同时达成 A 和 B 两个目标。

$$\frac{\Delta \Vdash A\ true \qquad \Delta' \Vdash B\ true}{\Delta, \Delta' \Vdash A \otimes B\ true}\ \otimes\mathrm{I}$$

$$\frac{\Delta \Vdash A \otimes B\ true \qquad \Delta', A\ true, B\ true \Vdash C\ true}{\Delta, \Delta' \Vdash C\ true}\ \otimes\mathrm{E}$$

我们把同时合取读作“A 且 B”。

[9] J.-Y. Girard. 1987. Linear logic. *Theor. Comp. Sci.*, 50, 1. DOI: 10.1016/0304-3975(87)90045-4.

线性逻辑

定义 (可选合取)

我们用 $A \,\&\, B$ 表示可以用一些资源达成目标 A，也可以达成目标 B，但是在消除时才确定要达成哪个。

$$\frac{\Delta \Vdash A \ true \qquad \Delta \Vdash B \ true}{\Delta \Vdash A \,\&\, B \ true}\ \&I$$

$$\frac{\Delta \Vdash A \,\&\, B \ true}{\Delta \Vdash A \ true}\ \&E_1 \qquad\qquad \frac{\Delta \Vdash A \,\&\, B \ true}{\Delta \Vdash B \ true}\ \&E_2$$

我们把可选合取读作“A 与 B”。

回顾

可选合取的引入和消除规则的形式与命题逻辑中合取的引入和消除规则是相同的。

线性逻辑

定义 (线性蕴含)

我们用 $A \multimap B$ 表示可以使用资源 A 来达成目标 B。

$$\frac{\Delta, A\ true \Vdash B\ true}{\Delta \Vdash A \multimap B\ true} \multimap I$$

$$\frac{\Delta \Vdash A \multimap B\ true \qquad \Delta' \Vdash A\ true}{\Delta, \Delta' \Vdash B\ true} \multimap E$$

例子 (砖块世界)

若机械臂为空,砖块 x 在 y 上面且 x 上面为空,则可以状态转移到机械臂抓着砖块 x,砖块 y 上面为空。

$$\forall x.\ \forall y.\ (\mathsf{empty} \otimes \mathsf{on}(x, y) \otimes \mathsf{clear}(x)) \multimap (\mathsf{holds}(x) \otimes \mathsf{clear}(y))$$

线性逻辑

定义 (零个命题的同时合取)

我们用 $\mathbf{1}$ 表示不用任何资源就能达成的目标。

$$\frac{}{\cdot \Vdash \mathbf{1}\ true}\ \mathbf{1}\mathrm{I}$$

$$\frac{\Delta \Vdash \mathbf{1}\ true \qquad \Delta' \Vdash C\ true}{\Delta, \Delta' \Vdash C\ true}\ \mathbf{1}\mathrm{E}$$

$\cdot \Vdash A \multimap (A \otimes \mathbf{1})\ true$

$$\dfrac{\dfrac{\dfrac{}{A\ true \Vdash A\ true}\ \mathsf{ID} \qquad \dfrac{}{\cdot \Vdash \mathbf{1}\ true}\ \mathbf{1}\mathrm{I}}{A\ true \Vdash A \otimes \mathbf{1}\ true}\ \otimes\mathrm{I}}{\cdot \Vdash A \multimap (A \otimes \mathbf{1})\ true}\ \multimap\mathrm{I}$$

线性逻辑

定义 (零个命题的可选合取)

我们用 $\top$ 表示可以在任何资源情况下达成的目标。换句话说，这表达了命题逻辑中的永真。

$$\frac{}{\Delta \Vdash \top\ true}\ \top\mathrm{I}$$

$\cdot \Vdash A \multimap (A \,\&\, \top)\ true$

$$\dfrac{\dfrac{\dfrac{}{A\ true \Vdash A\ true}\ \mathsf{ID} \qquad \dfrac{}{A\ true \Vdash \top\ true}\ \top\mathrm{I}}{A\ true \Vdash A \,\&\, \top\ true}\ \&\mathrm{I}}{\cdot \Vdash A \multimap (A \,\&\, \top)\ true}\ \multimap\mathrm{I}$$

线性逻辑

定义 (析取)

我们用 $A \oplus B$ 表示可以用一些资源达成目标 A，也可以达成目标 B，并在引入时明确达成哪个。

$$\frac{\Delta \Vdash A\ true}{\Delta \Vdash A \oplus B\ true}\ \oplus I_1 \qquad \frac{\Delta \Vdash B\ true}{\Delta \Vdash A \oplus B\ true}\ \oplus I_2$$

$$\frac{\Delta \Vdash A \oplus B\ true \qquad \Delta', A\ true \Vdash C\ true \qquad \Delta', B\ true \Vdash C\ true}{\Delta, \Delta' \Vdash C\ true}\ \oplus E$$

我们把析取读作“A 或 B”。

观察

这里的析取与命题逻辑中的析取相似，消除规则都是在进行分类讨论。

线性逻辑

定义 (零个命题的析取)

我们用 $\bot$ 表示不可能达成的目标。换句话说，这表达了命题逻辑中的永假。

$$\frac{\Delta \Vdash \bot \ \mathit{true}}{\Delta, \Delta' \Vdash C \ \mathit{true}}\ \bot\mathsf{E}$$

$\cdot \Vdash (A \oplus \bot) \multimap A \ \mathit{true}$

$$\dfrac{\dfrac{}{A \oplus \bot \ \mathit{true} \Vdash A \oplus \bot \ \mathit{true}}\ \mathsf{ID} \qquad \dfrac{}{A \ \mathit{true} \Vdash A \ \mathit{true}}\ \mathsf{ID} \qquad \dfrac{\dfrac{}{\bot \ \mathit{true} \Vdash \bot \ \mathit{true}}\ \mathsf{ID}}{\bot \ \mathit{true} \Vdash A \ \mathit{true}}\ \bot\mathsf{E}}{\dfrac{A \oplus \bot \ \mathit{true} \Vdash A \ \mathit{true}}{\cdot \Vdash (A \oplus \bot) \multimap A \ \mathit{true}}\ \multimap\mathsf{I}}\ \oplus\mathsf{E}$$

线性逻辑的柯里-霍华德同构

回顾

我们的口号是：

- 证明即是程序。
- 命题即是类型。
- 证明的归约即是程序的计算。

观察

直觉主义线性逻辑与(简单类型的)线性函数式编程间存在柯里-霍华德同构。

- 一个类型为 $A \multimap B$ 的函数在计算时只会使用它的参数恰好一次。
- 已被使用在函数式编程中支持状态变化[10],研究算法复杂度理论[11],分析程序的内存使用[12]。

[10] M. Hofmann. 2000. A Type System for Bounded Space and Functional In-Place Update–Extended Abstract. In *European Symp. on Programming* (ESOP'00), 165–179. DOI: 10.1007/3-540-46425-5_11.

[11] M. Hofmann. 1999. Linear types and non-size-increasing polynomial time computation. In *Logic in Computer Science* (LICS'99), 464–473. DOI: 10.1109/LICS.1999.782641.

[12] D. Walker and K. Watkins. 2001. On Regions and Linear Types (Extended Abstract). In *Int. Conf. on Functional Programming* (ICFP'01), 181–192. DOI: 10.1145/507635.507658; M. Hofmann and S. Jost. 2003. Static Prediction of Heap Space Usage for First-Order Functional Programs. In *Princ. of Prog. Lang.* (POPL'03), 185–197. DOI: 10.1145/604131.604148.

直觉主义线性逻辑的证明对象

$$\frac{}{u : A \Vdash u : A}\ \mathrm{ID}$$

$$\frac{\Delta \Vdash M : A \qquad \Delta' \Vdash N : B}{\Delta, \Delta' \Vdash M \otimes N : A \otimes B}\ \otimes\mathrm{I} \qquad \frac{\Delta \Vdash M : A \otimes B \qquad \Delta', u : A, w : B \Vdash N : C}{\Delta, \Delta' \Vdash \mathbf{let}\ u \otimes w = M\ \mathbf{in}\ N : C}\ \otimes\mathrm{E}$$

$$\frac{\Delta \Vdash M : A \qquad \Delta \Vdash N : B}{\Delta \Vdash \langle M, N \rangle : A \,\&\, B}\ \&\mathrm{I} \qquad \frac{\Delta \Vdash M : A \,\&\, B}{\Delta \Vdash \mathbf{fst}\ M : A}\ \&\mathrm{E}_1 \qquad \frac{\Delta \Vdash M : A \,\&\, B}{\Delta \Vdash \mathbf{snd}\ M : B}\ \&\mathrm{E}_2$$

$$\frac{\Delta, u : A \Vdash M : B}{\Delta \Vdash \lambda u.\ M : A \multimap B}\ \multimap\mathrm{I} \qquad \frac{\Delta \Vdash M : A \multimap B \qquad \Delta' \Vdash N : A}{\Delta, \Delta' \Vdash M\ N : B}\ \multimap\mathrm{E} \qquad \frac{}{\cdot \Vdash \star : \mathbf{1}}\ \mathbf{1}\mathrm{I}$$

$$\frac{\Delta \Vdash M : \mathbf{1} \qquad \Delta' \Vdash N : C}{\Delta, \Delta' \Vdash \mathbf{let}\ \star = M\ \mathbf{in}\ N : C}\ \mathbf{1}\mathrm{E} \qquad \frac{}{\Delta \Vdash \langle\rangle : \top}\ \top\mathrm{I} \qquad \frac{\Delta \Vdash M : A}{\Delta \Vdash \mathbf{inl}\ M : A \oplus B}\ \oplus\mathrm{I}_1 \qquad \frac{\Delta \Vdash M : B}{\Delta \Vdash \mathbf{inr}\ M : A \oplus B}\ \oplus\mathrm{I}_2$$

$$\frac{\Delta \Vdash M : A \oplus B \qquad \Delta', u : A \Vdash N : C \qquad \Delta', w : B \Vdash P : C}{\Delta, \Delta' \Vdash \mathbf{case}\ M\ (\mathbf{inl}\ u \Rightarrow N \mid \mathbf{inr}\ w \Rightarrow P) : C}\ \oplus\mathrm{E} \qquad \frac{\Delta \Vdash M : \bot}{\Delta, \Delta' \Vdash \mathbf{abort}\ M : C}\ \bot\mathrm{E}$$

提纲

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

改变推理方法

观察

人们发现自然演绎风格的推理过程中混合了正向和反向的思维方式，不太适合进行证明搜索。回顾

$$\frac{\Delta_1 \Vdash A \multimap B \ \mathit{true} \qquad \Delta_2 \Vdash A \ \mathit{true}}{\Delta_1, \Delta_2 \Vdash B \ \mathit{true}} \multimap E$$

如果目标是证明 $\Delta \Vdash B \ \mathit{true}$，而我们希望应用规则 $\multimap E$，那么我们需要猜测命题 A。

原则 (相继式演算 Sequent Calculus[13])

引入新的判断形式 $\Delta \Longrightarrow A$，它依然表达使用 Δ 中的资源可以达成目标 A，不过

- 推理规则都是自底向上的。
- 自然演绎的引入规则对应右规则，即连结词出现在右规则的结论中的 A 处。
- 自然演绎的消除规则对应左规则，即连结词出现在左规则的结论中的 Δ 处。

[13] G. Gentzen. 1935. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39, 176–210. English translation in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131, North-Holland, 1969. DOI: 10.1007/BF01201353.

同时合取

定义 (自然演绎中同时合取的引入与消除)

$$\frac{\Delta \Vdash A\ \mathit{true} \qquad \Delta' \Vdash B\ \mathit{true}}{\Delta, \Delta' \Vdash A \otimes B\ \mathit{true}}\ \otimes I \qquad \frac{\Delta \Vdash A \otimes B\ \mathit{true} \qquad \Delta', A\ \mathit{true}, B\ \mathit{true} \Vdash C\ \mathit{true}}{\Delta, \Delta' \Vdash C\ \mathit{true}}\ \otimes E$$

定义 (同时合取的右规则与左规则)

$$\frac{\Delta \Longrightarrow A \qquad \Delta' \Longrightarrow B}{\Delta, \Delta' \Longrightarrow A \otimes B}\ \otimes R \qquad \frac{\Delta, A, B \Longrightarrow C}{\Delta, A \otimes B \Longrightarrow C}\ \otimes L$$

可选合取

定义 (自然演绎中可选合取的引入与消除)

$$\frac{\Delta \Vdash A\ true \qquad \Delta \Vdash B\ true}{\Delta \Vdash A \,\&\, B\ true}\ \&I \qquad \frac{\Delta \Vdash A \,\&\, B\ true}{\Delta \Vdash A\ true}\ \&E_1 \qquad \frac{\Delta \Vdash A \,\&\, B\ true}{\Delta \Vdash B\ true}\ \&E_2$$

定义 (可选合取的右规则与左规则)

$$\frac{\Delta \Longrightarrow A \qquad \Delta \Longrightarrow B}{\Delta \Longrightarrow A \,\&\, B}\ \&R \qquad \frac{\Delta, A \Longrightarrow C}{\Delta, A \,\&\, B \Longrightarrow C}\ \&L_1 \qquad \frac{\Delta, B \Longrightarrow C}{\Delta, A \,\&\, B \Longrightarrow C}\ \&L_2$$

析取

定义 (自然演绎中析取的引入与消除)

$$\frac{\Delta \Vdash A\ \mathit{true}}{\Delta \Vdash A \oplus B\ \mathit{true}}\ \oplus I_1 \qquad \frac{\Delta \Vdash B\ \mathit{true}}{\Delta \Vdash A \oplus B\ \mathit{true}}\ \oplus I_2$$

$$\frac{\Delta \Vdash A \oplus B\ \mathit{true} \qquad \Delta', A\ \mathit{true} \Vdash C\ \mathit{true} \qquad \Delta', B\ \mathit{true} \Vdash C\ \mathit{true}}{\Delta, \Delta' \Vdash C\ \mathit{true}}\ \oplus E$$

定义 (析取的右规则与左规则)

$$\frac{\Delta \Longrightarrow A}{\Delta \Longrightarrow A \oplus B}\ \oplus R_1 \qquad \frac{\Delta \Longrightarrow B}{\Delta \Longrightarrow A \oplus B}\ \oplus R_2 \qquad \frac{\Delta, A \Longrightarrow C \qquad \Delta, B \Longrightarrow C}{\Delta, A \oplus B \Longrightarrow C}\ \oplus L$$

例子

$A \oplus (B \,\&\, C) \Longrightarrow (A \oplus B) \,\&\, (A \oplus C)$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \Longrightarrow A}\,\mathrm{ID}}{A \Longrightarrow A \oplus B}\,\oplus R_1 \qquad \dfrac{\dfrac{}{A \Longrightarrow A}\,\mathrm{ID}}{A \Longrightarrow A \oplus C}\,\oplus R_1}{A \Longrightarrow (A \oplus B) \,\&\, (A \oplus C)}\,\&R \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{B \Longrightarrow B}\,\mathrm{ID}}{B \Longrightarrow A \oplus B}\,\oplus R_2}{B \,\&\, C \Longrightarrow A \oplus B}\,\&L_1 \qquad \dfrac{\dfrac{\dfrac{}{C \Longrightarrow C}\,\mathrm{ID}}{C \Longrightarrow A \oplus C}\,\oplus R_2}{B \,\&\, C \Longrightarrow A \oplus C}\,\&L_2}{B \,\&\, C \Longrightarrow (A \oplus B) \,\&\, (A \oplus C)}\,\&R}{A \oplus (B \,\&\, C) \Longrightarrow (A \oplus B) \,\&\, (A \oplus C)}\,\oplus L$$

相继式演算：其余规则

$$\frac{}{A \Longrightarrow A}\ \mathrm{ID} \qquad \frac{\Delta \Longrightarrow A \qquad \Delta', A \Longrightarrow C}{\Delta, \Delta' \Longrightarrow C}\ \mathrm{CUT}$$

$$\frac{\Delta, A \Longrightarrow B}{\Delta \Longrightarrow A \multimap B}\ \multimap\mathrm{R} \qquad \frac{\Delta \Longrightarrow A \qquad \Delta', B \Longrightarrow C}{\Delta, \Delta', A \multimap B \Longrightarrow C}\ \multimap\mathrm{L}$$

$$\frac{}{\cdot \Longrightarrow \mathbf{1}}\ \mathbf{1}\mathrm{R} \qquad \frac{\Delta \Longrightarrow C}{\Delta, \mathbf{1} \Longrightarrow C}\ \mathbf{1}\mathrm{L} \qquad \frac{}{\Delta \Rightarrow \top}\ \top\mathrm{R} \qquad \frac{}{\Delta, \bot \Longrightarrow C}\ \bot\mathrm{L}$$

观察

- ID 规则和 CUT 规则不是必需的：实际上，它们是相继式演算系统的定理。
- 在报告中，我们将重点考察 CUT 规则，并称之为切割规则。

提纲

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

回顾

原则 (和谐性)

在一个自然演绎逻辑系统中，我们希望消除规则既不会太强，也不会太弱。

- 不会太强：对一个连结词应用消除规则不会获得比引入该连结词更多的信息。
- 不会太弱：对一个连结词应用消除规则后可以在得出的信息上重新对它进行证明。

让我们来从全局的角度考虑所有的左规则不会太强。回顾切割规则（即，切割定理）：

$$\frac{\Delta \Longrightarrow A \qquad \Delta', A \Longrightarrow C}{\Delta, \Delta' \Longrightarrow C}\ \text{CUT}$$

也就是说，我们在 $\Delta \Longrightarrow A$ 中应用一系列右规则证明了 A，然后在 $\Delta', A \Longrightarrow C$ 中应用一系列左规则来使用 A，这样并没有使得相继式演算获得额外的能力。

原则

切割定理从全局的角度说明了左规则不会太强。

切割归约

定义 (切割归约)

我们用如下记号表示从推理 $\mathcal{D}$ 到 $\mathcal{D}'$ 的切割归约，其中两个推理的结论都是 $\Delta \Longrightarrow C$，归约表示我们把一个使用切割规则且它的两个前提是同一命题的右和左规则的证明进行了化简：

$$\begin{array}{c}\mathcal{D}\\ \Delta \Longrightarrow C\end{array} \quad \rightsquigarrow_{R} \quad \begin{array}{c}\mathcal{D}'\\ \Delta \Longrightarrow C\end{array}$$

也就是说，切割归约的左边的推理模式如下：

$$\dfrac{\dfrac{\cdots}{\Delta \Longrightarrow A}\ A \text{ 的右规则} \qquad \dfrac{\cdots}{\Delta', A \Longrightarrow C}\ A \text{ 的左规则}}{\Delta, \Delta' \Longrightarrow C}\ \mathsf{CUT}_A$$

亚单例逻辑 Subsingleton Logic[14]

定义

考虑一个简化的线性逻辑的相继式演算：至多允许一个前件。
我们记为 $\delta \Longrightarrow C$，这里的 δ 表示零个或一个前件。

$$\frac{}{A \Longrightarrow A}\ \mathrm{ID} \qquad \frac{\delta \Longrightarrow A \qquad A \Longrightarrow C}{\delta \Longrightarrow C}\ \mathrm{CUT}$$

$$\frac{\delta \Longrightarrow A \qquad \delta \Longrightarrow B}{\delta \Longrightarrow A \,\&\, B}\ \&\mathrm{R} \qquad \frac{A \Longrightarrow C}{A \,\&\, B \Longrightarrow C}\ \&\mathrm{L}_1 \qquad \frac{B \Longrightarrow C}{A \,\&\, B \Longrightarrow C}\ \&\mathrm{L}_2 \qquad \frac{}{\cdot \Longrightarrow \mathbf{1}}\ \mathbf{1}\mathrm{R} \qquad \frac{\cdot \Longrightarrow C}{\mathbf{1} \Longrightarrow C}\ \mathbf{1}\mathrm{L}$$

$$\frac{\delta \Longrightarrow A}{\delta \Longrightarrow A \oplus B}\ \oplus\mathrm{R}_1 \qquad \frac{\delta \Longrightarrow B}{\delta \Longrightarrow A \oplus B}\ \oplus\mathrm{R}_2 \qquad \frac{A \Longrightarrow C \qquad B \Longrightarrow C}{A \oplus B \Longrightarrow C}\ \oplus\mathrm{L}$$

[14] H. DeYoung and F. Pfenning. 2016. Substructural Proofs as Automata. In *Asian Symp. on Prog. Lang. and Systems* (APLAS'16), 3–22. DOI: 10.1007/978-3-319-47958-3_1.

析取的切割归约

$\oplus R_1$ 和 $\oplus L$

$$\dfrac{\dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow A}}{\delta \Longrightarrow A \oplus B}\oplus R_1 \qquad \dfrac{\dfrac{\mathcal{E}}{A \Longrightarrow C} \qquad \dfrac{\mathcal{F}}{B \Longrightarrow C}}{A \oplus B \Longrightarrow C}\oplus L}{\delta \Longrightarrow C}\,\mathsf{CUT}_{A\oplus B} \quad \rightsquigarrow_R \quad \dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow A} \qquad \dfrac{\mathcal{E}}{A \Longrightarrow C}}{\delta \Longrightarrow C}\,\mathsf{CUT}_A$$

$\oplus R_2$ 和 $\oplus L$

$$\dfrac{\dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow B}}{\delta \Longrightarrow A \oplus B}\oplus R_2 \qquad \dfrac{\dfrac{\mathcal{E}}{A \Longrightarrow C} \qquad \dfrac{\mathcal{F}}{B \Longrightarrow C}}{A \oplus B \Longrightarrow C}\oplus L}{\delta \Longrightarrow C}\,\mathsf{CUT}_{A\oplus B} \quad \rightsquigarrow_R \quad \dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow B} \qquad \dfrac{\mathcal{F}}{B \Longrightarrow C}}{\delta \Longrightarrow C}\,\mathsf{CUT}_B$$

可选合取的切割归约

&R 和 &L$_1$

$$\dfrac{\dfrac{\begin{array}{cc}\mathcal{D} & \mathcal{E}\\ \delta \Longrightarrow A & \delta \Longrightarrow B\end{array}}{\delta \Longrightarrow A \,\&\, B}\,\&R \qquad \dfrac{\mathcal{F}}{A \,\&\, B \Longrightarrow C}\,\&L_1}{\delta \Longrightarrow C}\,\mathsf{CUT}_{A\&B} \quad \rightsquigarrow_R \quad \dfrac{\begin{array}{cc}\mathcal{D} & \mathcal{F}\\ \delta \Longrightarrow A & A \Longrightarrow C\end{array}}{\delta \Longrightarrow C}\,\mathsf{CUT}_A$$

&R 和 &L$_2$

$$\dfrac{\dfrac{\begin{array}{cc}\mathcal{D} & \mathcal{E}\\ \delta \Longrightarrow A & \delta \Longrightarrow B\end{array}}{\delta \Longrightarrow A \,\&\, B}\,\&R \qquad \dfrac{\mathcal{F}}{A \,\&\, B \Longrightarrow C}\,\&L_2}{\delta \Longrightarrow C}\,\mathsf{CUT}_{A\&B} \quad \rightsquigarrow_R \quad \dfrac{\begin{array}{cc}\mathcal{E} & \mathcal{F}\\ \delta \Longrightarrow B & B \Longrightarrow C\end{array}}{\delta \Longrightarrow C}\,\mathsf{CUT}_B$$

$\mathbf{1}$ 的切割归约

$\mathbf{1}\mathrm{R}$ 和 $\mathbf{1}\mathrm{L}$

$$\dfrac{\dfrac{}{\cdot \Longrightarrow \mathbf{1}}\,\mathbf{1}\mathrm{R} \qquad \dfrac{\begin{array}{c}\mathcal{D}\\ \cdot \Longrightarrow \mathrm{C}\end{array}}{\mathbf{1} \Longrightarrow \mathrm{C}}\,\mathbf{1}\mathrm{L}}{\cdot \Longrightarrow \mathrm{C}}\,\mathrm{CUT}_{\mathbf{1}} \quad \rightsquigarrow_{\mathrm{R}} \quad \begin{array}{c}\mathcal{D}\\ \cdot \Longrightarrow \mathrm{C}\end{array}$$

提纲

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

柯里-霍华德同构

逻辑系统		编程语言
线性逻辑命题	$\Longleftrightarrow$	类型??
相继式演算证明	$\Longleftrightarrow$	程序??
证明的切割归约	$\Longleftrightarrow$	程序的计算??

原则

在这一部分我们将看到,如上同构关系下的编程模型是基于消息传递的并发计算。

证明归约：析取

定义 (析取的切割归约之一)

$$\dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow A}}{\dfrac{\delta \Longrightarrow A \oplus B}{\delta \Longrightarrow C}}\oplus R_1 \quad \dfrac{\dfrac{\mathcal{E}}{A \Longrightarrow C} \quad \dfrac{\mathcal{F}}{B \Longrightarrow C}}{A \oplus B \Longrightarrow C}\oplus L \quad \text{CUT}_{A\oplus B} \quad \rightsquigarrow_R \quad \dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow A} \quad \dfrac{\mathcal{E}}{A \Longrightarrow C}}{\delta \Longrightarrow C}\text{CUT}_A$$

观察

- 归约前后的切割规则的前提中，都涉及了两个证明。
- 归约之后的切割规则的前提中，两个证明都发生了简化。
- “证明即是程序”：切割归约代表我们对两个程序同时进行了计算！

证明归约：析取

定义 (析取的证明归约之一)

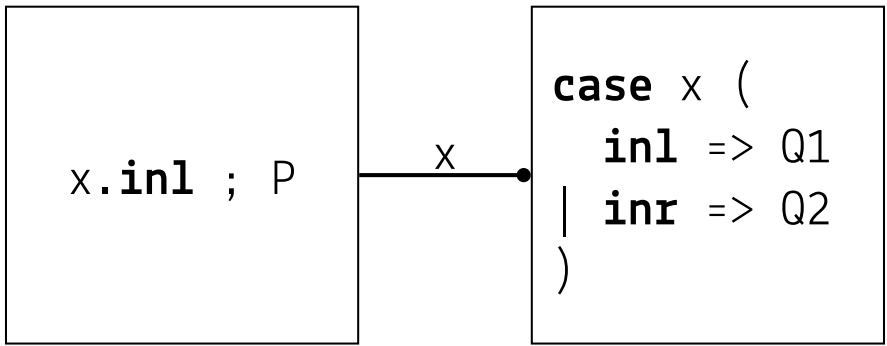
$$\dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow P :: (x : A)}}{\delta \Longrightarrow (x.\mathbf{inl}\ ;\ P) :: (x : A \oplus B)}\ \oplus R_1 \qquad \dfrac{\mathcal{E} \qquad \mathcal{F}}{(x : A \oplus B) \Longrightarrow \mathbf{case}\ x\ (\mathbf{inl} \Rightarrow Q_1 \mid \mathbf{inr} \Rightarrow Q_2) :: (z : C)}\ \oplus L}{\delta \Longrightarrow ((x.\mathbf{inl}\ ;\ P) \parallel \mathbf{case}\ x\ (\mathbf{inl} \Rightarrow Q_1 \mid \mathbf{inr} \Rightarrow Q_2)) :: (z : C)}\ \mathsf{CUT}_{A \oplus B}$$

$$\rightsquigarrow_R \quad \dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow P :: (x : A)} \qquad \dfrac{\mathcal{E}}{(x : A) \Longrightarrow Q_1 :: (z : C)}}{\delta \Longrightarrow (P \parallel Q_1) :: (z : C)}\ \mathsf{CUT}_A$$

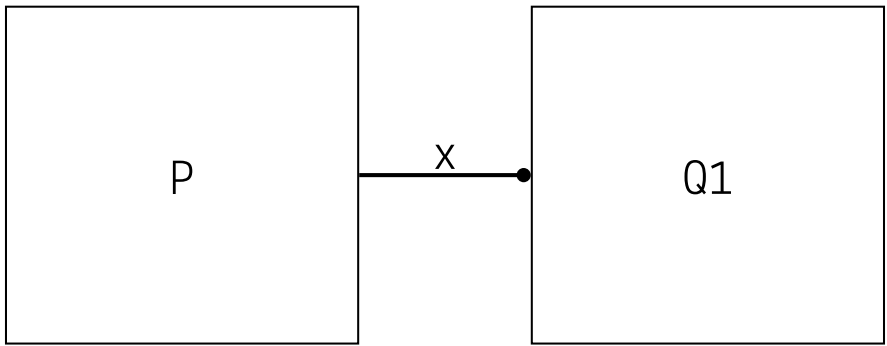
原则

- $(x : A)$ 表示为“信道 x 上有可以描述为类型 A 的消息传递”。
- $x.\mathbf{inl}$ 表示“在信道 x 上发送消息 **inl**”。**case** x 表示“从信道 x 上接收消息”。

基于消息传递的并发计算



基于消息传递的并发计算



证明归约：可选合取

定义 (可选合取的证明归约之一)

$$\dfrac{\dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow P_1 :: (x : A)} \qquad \dfrac{\mathcal{E}}{\delta \Longrightarrow P_2 :: (x : B)}}{\delta \Longrightarrow \mathbf{case}\ x\ (\mathbf{inl} \Rightarrow P_1 \mid \mathbf{inr} \Rightarrow P_2) :: (x : A \,\&\, B)}\,\&R \qquad \dfrac{\mathcal{F}}{(x : A) \Longrightarrow Q :: (z : C)}}{(x : A \,\&\, B) \Longrightarrow (x.\mathbf{inl}\ ;\ Q) :: (z : C)}\,\&L_1}{\delta \Longrightarrow (\mathbf{case}\ x\ (\mathbf{inl} \Rightarrow P_1 \mid \mathbf{inr} \Rightarrow P_2) \parallel (x.\mathbf{inl}\ ;\ Q)) :: (z : C)}\,CUT_{A\&B}$$

$$\rightsquigarrow_R \quad \dfrac{\dfrac{\mathcal{D}}{\delta \Longrightarrow P_1 :: (x : A)} \qquad \dfrac{\mathcal{F}}{(x : A) \Longrightarrow Q :: (z : C)}}{\delta \Longrightarrow P_1 \parallel Q :: (z : C)}\,CUT_A$$

观察

析取和可选合取是对偶的：一个从左往右发消息，一个从右往左发消息。

证明归约：1

定义 (1 的证明归约)

$$\dfrac{\dfrac{}{\cdot \Longrightarrow \mathbf{close}\ x :: (x : \mathbf{1})}\,\mathbf{1}R \qquad \dfrac{\mathcal{D}}{\cdot \Longrightarrow Q :: (z : C)}}{\dfrac{}{(x : \mathbf{1}) \Longrightarrow (\mathbf{wait}\ x\ ;\ Q) :: (z : C)}\,\mathbf{1}L}\ \mathsf{CUT}_{\mathbf{1}} \quad \rightsquigarrow_R \quad \dfrac{\mathcal{D}}{\cdot \Longrightarrow Q :: (z : C)}$$

原则

- **close** x 表示关闭信道 x 并终止计算。
- **wait** x 则表示等待信道 x 关闭。

会话类型 Session Types[16]

上述用类型表示信道上的消息传递的方法被称为会话类型。
其与直觉主义线性逻辑的柯里-霍华德同构由 Caires 和 Pfenning 发现并形式化[15]。

定理

在基于(直觉主义)会话类型的并发计算中,所有的信息发送和接收都是匹配的,且不会发生死锁。

逻辑系统		编程语言
线性逻辑命题	$\Longleftrightarrow$	会话类型
相继式演算证明	$\Longleftrightarrow$	消息传递的并发程序
证明的切割归约	$\Longleftrightarrow$	程序的并发计算

[15] L. Caires and F. Pfenning. 2010. Session Types as Intuitionistic Linear Propositions. In *Int. Conf. on Concurrency Theory* (CONCUR'10). DOI: 10.1007/978-3-642-15375-4_16.

[16] K. Honda. 1993. Types for Dyadic Interaction. In *Int. Conf. on Concurrency Theory* (CONCUR'93). DOI: 10.1007/3-540-57208-2_35.

提纲

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

我的最近工作：Probabilistic Resource-Aware Session Types[17]

观察

- 在基于消息传递的并发计算中，我们想要对消息发送/接收的概率分布进行分析。
- 该工作的设计主要是从编程语言的角度出发的，对其逻辑基础没有深入研究。

定义

回顾直觉主义线性逻辑中基于资源和目标的解释，考虑

- 新命题 $A \oplus_p B$，其中 $p \in [0,1]$，表示“以 p 的概率达成目标 A，$1-p$ 的概率达成目标 B”（引入时）。
- 新命题 $A \mathbin{\&}_p B$，其中 $p \in [0,1]$，表示“以 p 的概率达成目标 A，$1-p$ 的概率达成目标 B”（消除时）。

[17] A. Das, D. Wang, and J. Hoffmann. 2023. Probabilistic Resource-Aware Session Types. *Proc. ACM Program. Lang.*, 7, 66, 1925–1956, POPL. DOI: 10.1145/3571259.

线性逻辑 + 概率

$$\frac{\delta \Longrightarrow A}{\delta \Longrightarrow A \oplus_1 B}\ \oplus_P R_1 \qquad \frac{\delta \Longrightarrow B}{\delta \Longrightarrow A \oplus_0 B}\ \oplus_P R_2$$

$$\frac{A \Longrightarrow C_1 \qquad B \Longrightarrow C_2 \qquad C = p \cdot C_1 +^R (1-p) \cdot C_2}{A \oplus_p B \Longrightarrow C}\ \oplus_P L$$

定义 (命题的概率组合)

考虑 $p, q_1, q_2 \in [0,1]$。

$$p \cdot (A \oplus_{q_1} B) +^R (1-p) \cdot (A \oplus_{q_2} B) \stackrel{\text{def}}{=} A \oplus_{p \cdot q_1 + (1-p) \cdot q_2} B$$

$$p \cdot C +^R (1-p) \cdot C \stackrel{\text{def}}{=} C$$

例子

对任意 $p \in [0,1]$，有 $p \cdot (B \oplus_0 A) +^R (1-p) \cdot (B \oplus_1 A) = B \oplus_{1-p} A$。

线性逻辑 + 概率

$$\frac{\delta_1 \Longrightarrow A \qquad \delta_2 \Longrightarrow B \qquad \delta = p \cdot \delta_1 +^{L} (1-p) \cdot \delta_2}{\delta \Longrightarrow A \,\&_p\, B}\ \&_p R \qquad \frac{A \Longrightarrow C}{A \,\&_1\, B \Longrightarrow C}\ \&_p L_1$$

$$\frac{B \Longrightarrow C}{A \,\&_0\, B \Longrightarrow C}\ \&_p L_2$$

定义 (命题的概率组合)

考虑 $p, q_1, q_2 \in [0,1]$。

$$p \cdot (A \,\&_{q_1}\, B) +^{L} (1-p) \cdot (A \,\&_{q_2}\, B) \stackrel{\text{def}}{=} A \,\&_{p \cdot q_1 + (1-p) \cdot q_2}\, B$$

$$p \cdot C +^{L} (1-p) \cdot C \stackrel{\text{def}}{=} C$$

例子

$A \oplus_p B \Longrightarrow B \oplus_{1-p} A$

$$\dfrac{\dfrac{\dfrac{}{A \Longrightarrow A}\,\mathsf{ID}}{A \Longrightarrow B \oplus_0 A}\oplus_P R_2 \qquad \dfrac{\dfrac{}{B \Longrightarrow B}\,\mathsf{ID}}{B \Longrightarrow B \oplus_1 A}\oplus_P R_1 \qquad B \oplus_{1-p} A = p \cdot (B \oplus_0 A) +^R (1-p) \cdot (B \oplus_1 A)}{A \oplus_p B \Longrightarrow B \oplus_{1-p} A}\oplus_P L$$

$A \oplus_p A \Longrightarrow A$

$$\dfrac{\dfrac{}{A \Longrightarrow A}\,\mathsf{ID} \qquad \dfrac{}{A \Longrightarrow A}\,\mathsf{ID} \qquad A = p \cdot A +^R (1-p) \cdot A}{A \oplus_p A \Longrightarrow A}\oplus_P L$$

例子

$$A \oplus_p (B \,\&\, C) \Longrightarrow (A \oplus_p B) \,\&\, (A \oplus_p C)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \Longrightarrow A}\,\mathsf{ID}}{A \Longrightarrow A \oplus_1 B}\,\oplus_P R_1 \qquad \dfrac{\dfrac{\dfrac{}{B \Longrightarrow B}\,\mathsf{ID}}{B \,\&\, C \Longrightarrow B}\,\&L_1}{B \,\&\, C \Longrightarrow A \oplus_0 B}\,\oplus_P R_2}{A \oplus_p (B \,\&\, C) \Longrightarrow A \oplus_p B}\,\oplus_P L \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A \Longrightarrow A}\,\mathsf{ID}}{A \Longrightarrow A \oplus_1 C}\,\oplus_P R_1 \qquad \dfrac{\dfrac{\dfrac{}{C \Longrightarrow C}\,\mathsf{ID}}{B \,\&\, C \Longrightarrow C}\,\&L_2}{B \,\&\, C \Longrightarrow A \oplus_0 C}\,\oplus_P R_2}{A \oplus_p (B \,\&\, C) \Longrightarrow A \oplus_p C}\,\oplus_P L}{A \oplus_p (B \,\&\, C) \Longrightarrow (A \oplus_p B) \,\&\, (A \oplus_p C)}\,\&R$$

线性逻辑 + 概率：逻辑基础是什么？

问题

- 线性逻辑的判断形式是否适用于概率？有没有可能定义“有概率 p 可能达成目标 A”之类的命题？
- 新加入的规则保持和谐性吗？即左规则不会太强也不会太弱？
- 命题的概率组合的实质是什么？它的定义是否合理，即不会太强也不会太弱？
- 由切割归约导出的计算模型长什么样？原工作中的计算模型非常复杂（“多宇宙语义”）。
- 如果回归到自然演绎，同构出来的编程语言长什么样？
- …

总结

直觉主义命题逻辑

证明即是程序，命题即是类型

函数式编程

直觉主义模态逻辑的一种同构

直觉主义线性逻辑的一种同构

改变推理方法：相继式演算

改变归约方式：切割归约

直觉主义线性逻辑的另一种同构

线性逻辑 + 概率？

柯里-霍华德同构

逻辑系统		编程语言
命题	$\Longleftrightarrow$	类型
证明	$\Longleftrightarrow$	程序
证明的归约	$\Longleftrightarrow$	程序的计算

原则 (协同设计)

- 在设计编程语言时,同时研究其对应的逻辑系统。
- 这种协同设计有很多变量:判断的形式,推理的方法,归约的方式,相等的处理,...
- 虽然已有一些理论指导(如和谐性),但通常还是很有难度!

参考资料

Frank Pfenning 教授的逻辑课讲义

- **Constructive Logic**：http://www.cs.cmu.edu/~fp/courses/15317-s23/index.html
- **Modal Logic**：http://www.cs.cmu.edu/~fp/courses/15816-s10/index.html
- **Linear Logic**：https://www.cs.cmu.edu/~fp/courses/15816-f01/index.html
- **Substructural Logic**：http://www.cs.cmu.edu/~fp/courses/15816-f16/index.html

Stanford Encyclopedia of Philosophy

- **Intuitionism in the Philosophy of Mathematics**：https://plato.stanford.edu/entries/intuitionism/
- **Constructive Mathematics**：https://plato.stanford.edu/entries/mathematics-constructive/
- **Intuitionistic Logic**：https://plato.stanford.edu/entries/logic-intuitionistic/
- **Modal Logic**：https://plato.stanford.edu/entries/logic-modal/
- **Linear Logic**：https://plato.stanford.edu/entries/logic-linear/
- **Substructural Logics**：https://plato.stanford.edu/entries/logic-substructural/