11 λD 中 Flag 风格的自然演绎
Flag-style natural deduction in λD

读书笔记

许博

1 λD 中的形式化推导

在 λD 中，我们可以以更有效更优雅的方式表示逻辑，尤其是构造逻辑。在章节 7.1 和 7.2 中，我们遇到了一些 λC 中处理逻辑的“隐藏”定义。作为例子，使用 λrmD 的标准形式表示下列三者：

Absurdity

λC 使用 ⊥ 表示 $\Pi\alpha : *.\alpha$，行为与描述性定义相同，在 λD 中可以写作：

$\emptyset \rhd \perp() := \Pi\alpha : *.\alpha : *$。

Negation

之前使用 $\neg A$ 作为 $A \to \perp$ 的简写，同样也是一个描述性定义：

$A : * \rhd \neg(A) := A \to \perp() : *$。

Conjunction

同样的，我们可以定义合取，具有了两个参数：

$A : *, B : * \rhd \wedge(A, B) := \Pi C : *.(A \to B \to C) \to C : *$。

形如以上的逻辑定义在 λD 中能够被形式化推导。因为需要先推导出定义实体之后，才能够在环境中添加定义。比如 ¬ 的定义的一个 λD_0-推导：

$$
\begin{array}{lclcl}
\mathcal{D}_1 & \equiv & \emptyset & \rhd & \bot() := \Pi\alpha:*.\,\alpha : * \\
\mathcal{D}_2 & \equiv & A:* & \rhd & \neg(A) := A\to\bot() : *
\end{array}
$$

(1)	$\emptyset$	;	$\emptyset$	$\vdash$	$*$	:	$\square$	(*sort*)
(2)	$\emptyset$	;	$\alpha : *$	$\vdash$	α	:	$*$	(*var*) on (1)
(3)	$\emptyset$	;	$\emptyset$	$\vdash$	$\Pi\alpha:*.\,\alpha$	:	$*$	(*form*) on (1), (2)
(4)	$\mathcal{D}_1$	;	$\emptyset$	$\vdash$	$\bot()$	:	$*$	(*par*) on (3)
(5)	$\mathcal{D}_1$	;	$\emptyset$	$\vdash$	$*$	:	$\square$	(*def*) on (1), (3)
(6)	$\mathcal{D}_1$	;	$A : *$	$\vdash$	A	:	$*$	(*var*) on (5)
(7)	$\mathcal{D}_1$	;	$A : *$	$\vdash$	$\bot()$	:	$*$	(*weak*) on (4), (5)
(8)	$\mathcal{D}_1$	;	$A : *,\ y : A$	$\vdash$	$\bot()$	:	$*$	(*weak*) on (7), (6)
(9)	$\mathcal{D}_1$	;	$A : *$	$\vdash$	$A\to\bot()$	:	$*$	(*form*) on (6), (8)
(10)	$\mathcal{D}_1, \mathcal{D}_2$	;	$A : *$	$\vdash$	$\neg(A)$	:	$*$	(*par*) on (9)

Figure 11.2 A λD_0-derivation for the definition of $\neg$

通常来说，推导会从底端开始，也即需要考虑如果能够到达最后的推定。需要从推导规则，不断从 **conclusion** 反推所需要的 **premiss**。

2 对比形式化和 flag-风格的 λD

在第四章及以后，我们在推导时，将应用了规则 (sort),(var),(weak) 以及 (form) 的步骤省略，以获得更紧凑的推导过程，同时舍去了一些并不有趣的步骤，现在在 λD 中，也将这样做，除了上述提到的规则，还加入了规则 (def)。

一个有趣的问题是如何使用 flag-风格表示图 11.2 中的线性格式，如下（已经使用了简化版本）：

(3)	$\Pi\alpha:*.\,\alpha : *$	(*form*)
($\mathcal{D}_1$)	$\bot() := \Pi\alpha:*.\,\alpha : *$	definition
(4)	$\bot() : *$	(*par*) on (3) and ($\mathcal{D}_1$)
(a)	$\boxed{A : *}$	
(6)	$\quad A : *$	(*var*)
(b)	$\quad \boxed{y : A}$	
(8)	$\quad\quad \bot() : *$	(*weak*), twice, on (4) and (6)
(9)	$\quad A\to\bot() : *$	(*form*) on (6) and (8)
($\mathcal{D}_2$)	$\quad \neg(A) := A\to\bot() : *$	definition
(10)	$\quad \neg(A) : *$	(*par*) on (9) and ($\mathcal{D}_2$)

两个版本之间极为相似。而观察行 (3) 到 (4)，会发现信息的重复，以及 ($\mathcal{D}_1$) 和行 (9) 和 (10)。看起来只保留 ($\mathcal{D}_1$) 和 ($\mathcal{D}_2$) 而删除行 (3),(4),(9)

和 (10) 是非常合理的。

以及行 (6) 和 (8) 或多或少都有些多余，行 (9) 是假设 (a) 和行 (4) 的一个显而易见的结果，所以我们可以跳过行 (6) 和 (8)，以及假设 (b)。

应用了上述修改，得到如下 flag-风格的推导：

$(\mathcal{D}_1)$ $\perp() \ := \ \Pi\alpha : *.\ \alpha \ : \ *$ $\quad$ $(form)$ and (par)

(a) $\boxed{A : *}$

$(\mathcal{D}_2)$ $\quad \neg(A) \ := \ A \to \perp() \ : \ *$ $\quad$ $(form)$ and (par)

至此，通过重写证明到 flag 形式，以及省略一些非常明显的行，我们得到了一个 flag 风格的推导，与在章节 11.1 开头相对应的定义的表示相同。

3 关于 λD 中 flag-风格证明的惯例

具有如下格式的一行：

$c(\overline{x}) := M : N$

在一个环境 Δ 和上下文 Γ 中，其中参数列表 $\overline{x}$ 是 Γ 中主体变量的列表，具有三重含义：

(1) 语句 $M : N$ 是关联 Δ 和 Γ 可推导的，

(2) 定义 $\mathcal{D} \equiv \Gamma \rhd c(\overline{x}) := M : N$ 被添加在 Δ 的结尾，

(3) 语句 $c(\overline{x}) : N$ 在扩展后的环境 $\Delta, \mathcal{D}$ 和上下文 Γ 中是可推导的。

通过为 flag-风格的推导精确地添加“操作语义”是的这类的多重含义更形式化。在下边的例子中，记录一个推导的状态为一个环境和上下文的对 $\{\mathcal{D}|\Gamma\}$，这个状态在上下文改变以及一个新的定义被添加时变化：

$\{\emptyset \mid \emptyset\}$

(a) $\boxed{x_1 : A_1}$

$\{\emptyset \mid x_1 : A_1\}$

(b) $\boxed{x_2 : A_2}$

$\{\emptyset \mid x_1 : A_1, x_2 : A_2\}$

(1) $a(x_1, x_2) \ := \ M_1 : N_1$

$\{x_1 : A_1, x_2 : A_2 \rhd a(x_1, x_2) \ := \ M_1 : N_1 \mid x_1 : A_1, x_2 : A_2\}$

$\{x_1 : A_1, x_2 : A_2 \rhd a(x_1, x_2) \ := \ M_1 : N_1 \mid x_1 : A_1\}$

(2) $b(x_1) \ := \ M_2 : N_2$

$\{x_1 : A_1, x_2 : A_2 \rhd a(x_1, x_2) \ := \ M_1 : N_1, \ \ x_1 : A_1 \rhd b(x_1) \ := \ M_2 : N_2 \mid x_1 : A_1\}$

在这个例子中，可以看到：

- 每当一个 flag 提出时，上下文被扩展；当这个 flag 结束的时候，上下文 Γ

也随之收缩；
- 在一个上下文下的定义 $a(\overline{x}) := M : N$ 表示相应的定义被添加到环境 Δ 中，其中 Γ 对应于 flag 上下文。

另外，语句 $M : N$ 以及 $a(\overline{x})$（隐含在 $\mathcal{D}$）中，对应于两个可推导的推定：$\Delta; \Gamma \vdash M : N$ 和 $\Delta, \mathcal{D}; \Gamma \vdash a(\overline{x}) : N$。

在构建推导时，我们逐渐构造一个环境 Δ。可以选择任意时候从头开始一个推导，“抛弃”之前的信息。另一种方法是将所有获得的推导压缩至一个大的总体的推导，可以简单地通过将新的推导与旧的连接起来完成，从而使得所有的推定，包括定义都保持“存活”（合法以及可获得的），而缺点是会引入一些多余的推定（尤其是定义）。

在构造一个巨大的连续的推导时，上下文会在行间增减，而环境只会越来越大，因为没有删除定义的规则。但是读者可以添加额外的规则以删除多余的定义（可以通过正式的规则证明可行）。

不删除定义可以被证明是正当的，一个环境可以被看作是事实的列表，其中的事实重要或可能重要，记录的作为证明的推定可以始终在之后的阶段中被使用。因此，环境也作用为我们成果的一类 log-book。并且存在一种自然的对应于在一个（逻辑或数学）书中构建的“知识”和提到的一个推导的 log-book。

为了流线化 flag 推导的表示，自此我们将使用定义格式（definition format）。也即表示推导为定义的列表。因此在 flag 推导中，写成 $\Gamma \triangleright c(\overline{x}) := M : N$ 而不是 $\Gamma \vdash M : N$。

而上述惯例可能会引入不重要的被定义的名字，即在定义之后就不再使用的名字。另外，这种方式还会使得系统 λD 简化，尤其是使用 flag 格式表示时，因为不再有语句：语句都被嵌入于定义中。