软件科学基础第一次习题课

Rocq 的回顾和扩展与直觉主义逻辑的深入解释

张书豪

2026 年 4 月 9 日

北京大学计算机学院

今天的两个小专题

前半：Rocq 的回顾与扩展

- `simpl` vs. `reflexivity`
- `injection` / `discriminate` / `inversion`
- tactic vs. proof term

后半：深入理解直觉主义逻辑

- BHK 解释
- Kripke 语义
- witness 与 proof object

内容安排

两部分彼此独立。前半段用三组**对比**来理解 Rocq 中证明的构造与检查；后半段再解释 Rocq 默认逻辑背后的构造性含义。

Rocq 的回顾与扩展

Rocq 部分：三组对比

1. `simpl` vs. `reflexivity`：比较“化简目标”过程中的强度。
2. `injection` / `discriminate` / `inversion`：比较三种“如何消化已有假设”的方式。
3. tactic vs. proof term：比较“交互式地做证明”和“直接把证明对象写出来”。

simpl 没动，但为什么还能证明等价？

```
Definition my_one := 1.
Lemma test_opaque : 1 = my_one.
Proof.
  simpl.
  reflexivity.
Qed.
```

现象

`simpl.` 之后，目标看起来仍是

$$1 = \texttt{my\_one}$$

核心区别

`simpl` 主要负责把目标**显示性地化简一点**；

`reflexivity` 则会为了判定两边是否**可转换**，继续做更强的规约检查。

所以即使 `simpl` 在显示上没把 `my_one` 展开，`reflexivity` 仍会按需展开它，并发现两边都能归约到 1。

化简过程中，哪些规约在起作用？

规约	含义	simpl	compute	unfold	reflexivity
β	函数调用化简	执行	执行	间接涉及	执行
ι	match 化简	条件性执行	执行	间接涉及	执行
δ	定义展开	很保守	很激进	只展开指定定义	按需展开
ζ	let 展开	执行	执行	不专门负责	执行

所以“simpl 没动”并不意味着“系统看不出两边相等”；很多时候是 reflexivity 在最后一步把可转换性检查做完了。

如何消化已有假设

tactic	更适合干什么
`injection`	利用**单射性**，从相同构造子的等式中推出参数相等
`discriminate`	利用**互斥性**，从不同构造子的等式中直接推出矛盾
`inversion`	综合利用单射性与互斥性，从假设中提取更完整的结构信息

- 它们的共同点：都不是在“分情况讨论目标”，而是在**消化一个已有假设**。
- 它们的区别：`injection` 和 `discriminate` 更局部、更专门；`inversion` 往往会综合利用构造子的单射性、互斥性以及归纳类型的结构。

一个最小例子：从等式假设中读信息

```
Inductive foo : Type :=
| A : nat -> foo
| B : foo.
```

- 若 $H_1 : A\,n = A\,m$，可用 `injection H1` 得到 $n = m$。
- 若 $H_2 : A\,n = B$，可用 `discriminate H2` 直接推出矛盾。
- 上面两种情况都可以直接尝试 `inversion Hx`：它会自动把“可推出的参数相等”或“该假设不可能成立”一起整理出来。

构建证明的两大路线

交互式风格

- 面向当前目标逐步操作
- 写法更接近“证明过程”
- 适合探索、试错、局部修改

proof term 风格

- 直接写出证明对应的项
- 写法更接近“函数本体”
- 更直接体现 Curry–Howard 对应

同一个证明对象的两种视角

这不是两套不同的逻辑，而是同一个证明对象的两种视角：tactic 在帮你构造 proof term，proof term 则是构造完成后的结果。

同一个定理的两种写法

交互式写法

```
Theorem imp_trans :
  forall P Q R : Prop,
  (P -> Q) -> (Q -> R) -> P -> R.
Proof.
  intros P Q R HPQ HQR HP.
  apply HQR.
  apply HPQ.
  exact HP.
Qed.
```

proof term 写法

```
Definition imp_trans_term :
  forall P Q R : Prop,
  (P -> Q) -> (Q -> R) -> P -> R :=
  fun P Q R HPQ HQR HP =>
    HQR (HPQ HP).
```

Curry-Howard **对应**

尝试执行 `Print imp_trans.` 命令，你就会发现这正对应了右边的写法！

反例：错误的 proof term 会被类型检查拒绝

```
Fail Definition imp_trans_bad :
  forall P Q R : Prop,
  (P -> Q) -> (Q -> R) -> P -> R :=
  fun P Q R HPQ HQR HP =>
    HQR HP.
```

Rocq 会在类型检查阶段直接拒绝它，因此“写 proof term”并不是随手写注释，而是在提交一个要被内核验证的对象。

小结

1. `simpl` 与 `reflexivity` 的对比说明：“化简一点”和“检查等式成立”不是一回事。
2. `injection`、`discriminate`、`inversion` 的对比说明：Rocq 不只会拆目标，也会从假设中恢复结构信息。
3. tactic 与 proof term 的对比说明：证明既是一个交互过程，也是一个可以被显式写出的对象。

直觉主义逻辑的深入解释

再探直觉主义逻辑

- 为什么直觉主义逻辑不把某些经典原则当作一般定理？
- 为什么在 Rocq 中，证明析取和存在命题时，总强调“给证据”“给 witness”？

立场说明

这里不是在说“经典逻辑错了”，而是在解释一种不同的“何时算作已经证明”的标准。

BHK 解释：把命题看成构造任务

命题形式	构造性解释
$A \wedge B$	给出 A 的证明和 B 的证明
$A \vee B$	明确选左边或右边，并附上相应证明
$A \to B$	给出一个把 A 的证明变成 B 的方法
$\exists x, P(x)$	给出具体对象 x 与 $P(x)$ 的证明
$\forall x, P(x)$	给出对任意 x 都可工作的统一方法

一句话直觉

证明不是只报告“真值”，而是交出**构造**、**程序**、**证据**、witness。这正是各位写 Rocq 证明时干的事情！

经典原则的分家：不只是排中律

原则	公式	直觉主义中的地位
无矛盾律	$\neg(P \wedge \neg P)$	接受
排中律	$P \vee \neg P$	不作为定理
双重否定消去	$\neg\neg P \to P$	不作为定理

- 直觉主义逻辑并不接受“每个命题都已被预先决定真假”。
- 不接受某个经典原则，不等于承认它为假（你在作业中已经证明了排中律与直觉主义逻辑的相容性[1]！）。
- 直觉主义想强调的是：**我们无法给出构造性的证明**。

[1]见 Logic 一章的 `excluded_middle_irrefutable` 一题。

Kripke 语义：把“真”看成信息状态

- **世界**（world）：一个时刻、一个知识状态、一个信息阶段；在**不带量词的命题层面**，可先理解成当前掌握了哪些证明 / 证据。
- **可达关系**：$w \leq v$ 表示从 w 走到 v 时，信息更多了。
- **强迫关系**：$w \Vdash A$ 读作“在世界 w 中，A 被强迫成立”，也就是：按照当前信息，A 已经有足够的构造性根据，可以被断言。
- **单调性**（persistence）：

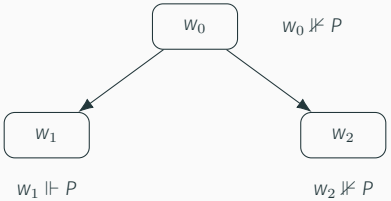
$$w \Vdash A \text{ 且 } w \leq v \implies v \Vdash A$$

- 一旦某命题已有构造性证据，未来不会把它“推翻”。

最重要的感觉
这里的不同世界不是“平行宇宙”，而是**信息多少不同的阶段**。

和经典语义的区别
经典语义更像“每个命题早就有真值”；Kripke 语义更像“当前能不能断言它成立，要看你现在手上有什么信息”。

最小反模型：排中律为何不在所有世界都成立？



用到联结词定义与模型中对 P 的指定
$w \Vdash A \vee B$ 意味着 $w \Vdash A$ 或 $w \Vdash B$；　　$w \Vdash \neg A$ 意味着对所有 $v \geq w$，都有 $v \nVdash A$。

- 在根世界 w_0，P 还不能断言。
- 但 $w_0 \nVdash \neg P$，因为存在后继世界 $w_1 \geq w_0$ 使得 $w_1 \Vdash P$。
- 因此在 w_0 里，P 和 $\neg P$ 都不能断言，所以 $w_0 \nVdash P \vee \neg P$。

结论：排中律失效不是说它的否定可证，而是说：存在 Kripke 模型，使它在某个世界里不被强迫。

带量词时，世界里的“信息”到底包括什么？

世界 w 中的信息可以这样理解
它不只记录“哪些命题已经有证明”，还记录：

- 当前**能谈论哪些对象**；
- 当前对这些对象**已经知道哪些事实 / 证据**；
- 当前是否已经有某个**统一方法**，能应付以后出现的对象。

- 所以从 w 走到更强的 v，信息增长可能同时表现为：对象更多了、关于对象的事实更多了、可用的证明更多了。
- 重点是：**信息不是只有对象域 D_w**。对象域只是世界里的一部分信息。
- 这也解释了为什么“在更强世界里能证明更多公式”并不等于“只是因为对象变多了”。

和不带量词的情形对比
命题层面时，世界大致可看成“当前已有的证明”；带量词时，更像“对象 + 关于对象的事实 + 证明”一起增长的语境。

对象域与候选值集合

对象域 D_w
表示在世界 w 里，**量词可以谈论哪些对象**。

- 现在知道 Alice、Bob
- 以后又认识 Carol

所以对象域可以增长。

候选值集合
表示某个未知量**当前还可能是谁**。

- **一开始** $y \in \{\text{Alice}, \text{Bob}, \text{Carol}\}$
- **后来缩成** $y \in \{\text{Alice}, \text{Carol}\}$

所以候选集可以缩小。

结论：量词语义用的是对象域 D_w，不是某个未知量的候选值集合。

量词的形式化定义与 witness

一阶 Kripke 模型中的数据
一个一阶 Kripke 模型除了世界与可达关系外，还带有

$$(D_w)_{w \in W} \quad \text{以及原子公式的强迫关系}$$

其中 W 是世界集合，D_w 是世界 w 的对象域，并满足

$$w \le v \Rightarrow D_w \subseteq D_v$$

原子公式也满足单调性：若 $w \Vdash P(a_1, \dots, a_n)$ 且 $w \le v$，则 $v \Vdash P(a_1, \dots, a_n)$。

- $w \Vdash \forall x\, \varphi(x) \iff \forall v \ge w\, \forall a \in D_v,\ v \Vdash \varphi(a)$
- $w \Vdash \exists x\, \varphi(x) \iff \exists a \in D_w,\ w \Vdash \varphi(a)$

为什么 witness 很关键？
$\neg\forall x\, \neg P(x) \to \exists x\, P(x)$ 在构造性读法里一般不成立：前件只说明没有统一反驳，后件却要求你**当前就拿出某个 witness**。

回到 Rocq：proof object、析取、存在

默认环境下

- 想证明 $P \vee \neg P$，通常做不到。
- 想证明 $\exists x, P(x)$，通常要真的给出某个 x。
- tactic 卡住时，常常不是“技术不够”，而是目标本身缺少构造信息。

加入经典公理后

- 许多经典结论会立刻变得可证。
- 但证明更像“逻辑上可推出”，不一定还能读出算法或 witness。
- 证明能力增强，构造内容往往减弱。

总结

默认的构造逻辑设置不是“故意为难人”，而是为了保留 proof object 与程序内容。

在 Rocq 中加入经典公理：一个最小例子

```
Theorem em_implies_dne :
  (forall P : Prop, P \/ ~ P) ->
  forall P : Prop, ~~P -> P.

Proof.
  intros EM P Hnn.
  destruct (EM P) as [HP | HnP].
  - exact HP.
  - exfalso. apply Hnn. exact HnP.
Qed.
```

小结

1. BHK 解释告诉我们：证明不是只说“真”，而是交出构造。
2. Kripke 世界表示**信息状态**，构造性真理满足**单调性**。
3. **构造逻辑中**，$\exists x, P(x)$ **真正要求** witness，**而不只是“无法统一否定”**。
4. **这也解释了为什么** Rocq **默认更关心证据**、witness、proof object。

Questions?