一阶逻辑的语法与语义

一个数学定理是由公理出发经过正确的逻辑推导得出的一个结论。我们希望这整个过程是尽可能精确而严格的，因此这整个过程最好是能够被形式化(formalize)的。数学的形式化一方面能够帮助检查数学自身的严格性，一方面也是在计算机上做数学证明的基础。为此我们要定义一套形式语言来描述公理、证明与定理。

一阶逻辑的语法(Syntax)

我们将会建立的这套形式语言称为一阶语言(first-order language)或一阶逻辑(first-order logic)。对于一个集合，我们把集合的元素(elements)称为一阶对象(first-order objects)，它们是构成集合的最基本要素。一阶对象的集合也即子集称为二阶对象(second-order objects)，相应地二阶对象的集合构成三阶对象，等等。一阶语言规定我们在量词中只能提到一阶对象，例如我们可以说“对于集合 A 中的每个元素，……”，但不能直接说“对于集合 A 的每个子集，……”。这意味着一阶逻辑能够直接表达的数学定理是有限的，有许多涉及高阶两次的数学定理是不能直接翻译为一阶逻辑的。然而，通过一些基于集合论的转化，我们原则上可以用一阶逻辑表达当今世界上的所有数学定理，也就是说本质上对一阶逻辑的讨论就是对所有数学定理的讨论，我们之后会看到如何做到这一点。

Alphabet

在形式语言中，一切讨论的对象都要用符号串(word)来表示，符号串就是由字母表(alphabet)中的字符连接而成的字符串。一阶逻辑的字母表中应当包含哪些字符呢？我们来考虑一些用自然语言表达的数学命题：一个群论中的命题是“群里存在一个元素使得所有元素和它相乘都等于自身”，我们看到这个命题中包含对元素的讨论，量词(quantifier)“存在”和“所有”，一个重要的概念“等于”，一个二元运算，也即一个“函数”；另一个表示等价关系的传递性的命题是“如果元素a和元素b等价并且元素b和元素c等价则元素a和元素c等价”，这个命题中包含一个因果推导和两个前提的并列的“逻辑关系”，用于表达两个元素等价的“二元关系”。一阶语言中还应当包括其它的一些表示“非”“或”“当且仅当”等等的符号。现在我们把这些符号总结如下：

一阶逻辑的alphabet包括以下内容：

- 变量(variables)：一般认为变量应当记为 $v_0, v_1, \cdots$，这里能够用自然数做下标，是因为我们默认一阶逻辑能够使用的变量总数是至多可数的（有时为了方便，也可以把变量记为 $x, y, z, \cdots$ 或 $a, b, c, \cdots$ 可数的）；
- 逻辑符号(logical symbols)：非（ $\neg$ ），与（ $\wedge$ ），或（ $\vee$ ），推出（ $\to$ ），等价（ $\leftrightarrow$ ）；
- 量词(quantifiers)：存在（ $\exists$ ），任意（ $\forall$ ）；
- 等号(equality symbol)：一阶逻辑中等号用（ $\equiv$ ）表示而不是算术中常用的（ $=$ ）；
- 括号(parentheses)：（ $($ ），（ $)$ ）。用来区分命题中各个部分的优先级；
- n 元关系符号(n -ary relation symbols)， $n \geq 1$ 。例如设 R 是一个二元关系符号，那么 Rv_0v_1 就表示一个二元关系，比如 v_0 与 v_1 等价；
- n 元函数符号(n -ary function symbols)， $n \geq 1$ 。例如设 f 是一个二元函数，那么 fv_0v_1 就表示一个二元函数的值，比如自然数 v_0 与 v_1 的和；
- 常数符号(constant symbols)， $c_0, c_1, \cdots$，或者也可以用特殊的字母表示。例如在群论中可以用 e 表示单位元这一常量。

我们注意到以上分类中的最后三类比较特殊。前面五类在所有场合都相同，而后三类则会在不同场合采用不同的符号，甚至如果讨论中不涉及的话可以为空。我们把这三类统称为符号集(symbol set)，记为 S 。前五类符号记为 $\mathcal{A}$ 。整个字母表记为 $\mathcal{A}_S$ 。所以事实上，我们认为我们在不同的场合会使用“不同”的字母表，这些不同的字母表只在符号集上有差异。（这里，场合是针对我们要形式化的对象的。例如如

果我们想用一阶逻辑形式化自然数上的定理，那么我们就可能需要二元关系“整除”，二元运算“加法”，以及常量“ 0 ”。而如果我们想要形式化群论上的定理，则不需要整除或加法这些符号，而需要群元素间的二元运算，一元函数“逆元”，常量“单位元”等等。只有确定要形式化的对象之后我们才能确定字母表，字母表确定了以后所有一阶逻辑写出的命题就只能包含字母表中的字符）

Terms & Formulas

根据一阶逻辑的字母表，我们可以把字母表中的字母任意排列连接成字符串，这些字符串可以是任意的，比如 $)v_0(\neg c_0\ \to\equiv$ 。然而并不是所有可能的组合形式的字符串都会是我们用到的，只有一小部分满足特定规则的字符串才会构成一阶逻辑的“语言”。和自然语言一样，形式语言也是需要有语法(syntax)的，语法就是那些会被我们使用的字符串要满足的构造规则。这样的构造规则是从人们日常使用的数学语言中抽象出来的。例如人们在学习二次方程的时候会用代数式与等号写出 $x^2+y^2=2xy$ ，等号两边的字符串称为“项”(term)，整个字符串称为“公式”(formula)。我们要抽象出像这样的特殊组合的构造规则，定义一阶逻辑的项和公式。

对于特定的符号集 S ，我们归纳地定义 S 下的一阶逻辑的项（称为 S -term）：

- 单个变量 v_i 是一个 S -term；
- 单个常数 c_i 是一个 S -term；
- 对于 $n\geq 1$ ，如果 $t_1,\cdots,t_n$ 都是 S -term，那么对于 n 元函数 f ， $ft_1t_2\cdots t_n$ 也是一个 S -term；

基于term的定义，我们归纳地定义 S 下的一阶逻辑的公式（称为 S -formula）：

- 如果 t_1 和 t_2 是两个 S -term，那么 $t_1\equiv t_2$ 是一个 S -formula；
- 对于 $n\geq 1$ ，如果 $t_1,\cdots,t_n$ 都是 S -term，那么对于 n 元关系 R ， $Rt_1t_2\cdots t_n$ 也是一个 S -formula；
- 如果 φ 是 S -formula， x 是一个变量， $\neg\varphi$ 、 $\forall x\varphi$ 、 $\exists x\varphi$ 也是 S -formula；
- 如果 φ,ψ 是 S -formula， $(\varphi\wedge\psi)$ 、 $(\varphi\vee\psi)$ 、 $(\varphi\to\psi)$ 、 $(\varphi\leftrightarrow\psi)$ 也是 S -formula。

应当指出，这里的“公式”与日常语言中的公式概念不同，这里的公式不是指恒成立的等式，而其实是一阶逻辑的“命题”。一阶逻辑的语言指的就是所有一阶逻辑的命题。所以，我们把符号集 S 下全体 S -formula的集合记为 L^S ，这里的 L 就是指language。全体 S -term的集合也有一个记号 T^S 。

特别重要的一点是，我们要求一阶逻辑的term和formula都是有限长的字符串。也就是说，我们要求上面的归纳定义中，归纳次数是有限次。于是，每个term或formula都可以从某个原子性的(atomic) term或formula出发一步步构造得到，构造的过程可以写作一个有限长的字符串序列。我们把这样的归纳称为结构归纳(structural induction)。

我们可以基于结构归纳定义关于term或formula的函数。一个重要的基于结构归纳的函数用来指明term中出现的所有变量的集合，这个函数可以定义如下：

- $\mathrm{var}(v_i):=v_i$ ；
- $\mathrm{var}(c_i):=\varnothing$ ；
- $\mathrm{var}(ft_1\cdots t_n):=\mathrm{var}(t_1)\cup\cdots\cup\mathrm{var}(t_n)$ ；

Sentences

在数学语言中，出现在量词后面的变量与一般的变量之间是不同的。例如，形式串 $\forall v_0\exists v_1\neg v_0\equiv v_1$ 与 $fv_0v_1=v_2$ 中变量试图表达的数学含义是不同的（尽管单从一阶逻辑的语法中我们不能看出任何“意义”，但根据以往我们对数学语言接触的经验我们可以模糊地猜出这两个形式串所试图抽象出来的含义），前者把变量当作一种临时的实例化来使用，后者则特定的指明某些个变量满足什么性质。这有点类似程序语言中的局部变量与全局变量，在一阶逻辑中我们把前者称为受限变量(bound variables)，后

者称为自由变量(free variables)。自由变量是formula当中不在量词中出现的那些变量，它是一个关于formula的函数，根据结构归纳定义如下：

- $\mathrm{free}(t_1 \equiv t_2) := \mathrm{var}(t_1) \cup \mathrm{var}(t_2)$；
- $\mathrm{free}(Rt_1 \cdots t_n) := \mathrm{var}(t_1) \cup \cdots \cup \mathrm{var}(t_n)$；
- $\mathrm{free}(\neg\varphi) := \mathrm{free}(\varphi)$
- $\mathrm{free}((\varphi * \psi)) := \mathrm{free}(\varphi) \cup \mathrm{free}(\psi)$，其中 $*$ 表示 $\wedge$, $\vee$, $\to$, $\iff$；
- $\mathrm{free}(\forall x\varphi) := \mathrm{free}(\varphi) \setminus x$
- $\mathrm{free}(\exists x\varphi) := \mathrm{free}(\varphi) \setminus x$

我们注意到根据定义，$\mathrm{free}((Ryx \to \forall y\neg y \equiv z)) = x, y, z$，即便 y 曾出现在某个 $\forall$ 后过。这是合理的，因为如果某个变量在小的层面是临时的，如果它又在更大的层面充当全局的，则应当被理解为是全局的。这就好像程序语言中局部变量可以与全局变量重名，但不影响全局变量依然具有全局性。

如果一个formula中没有自由变量，也即如果所有变量都是局部的，那么这个formula一定描述了一些重要的性质，这性质不依赖于某些特定的变量，而是所讨论的数学对象本身的性质。比如“群中的任意元素总是存在逆元”，这是群这一代数结构本身的特性，而与群中某几个元素之间的关系如何无关。我们把没有自由变量的formula称为一个sentence。通常，我们会把公式集 L^S 中只包含 $v_0, \cdots, v_{n-1}$ 作为自由变量的子集记为 L_n^S，$L_n^S := \varphi \mid \varphi \in L_n^S \text{且} \mathrm{free}(\varphi) \subseteq v_0, \cdots, v_n$，这样所有sentence的集合就可以记为 L_0^S。Sentence是一个很重要的概念，我们在接下来讨论语义的时候还会再次遇到。

一阶逻辑的语义(Semantics)

根据一阶逻辑的语法，我们能写出许多term和formula。到目前为止，它们都只是满足一定规则的字符串，而不具有“意义”。尽管这些字符串与我们在具体的数学中见到的项与公式很像，我们在看到一个一阶逻辑的项与公式时总能大致猜测出它试图表达的意义，但这种意义是未被定义的。在特定的场合下，我们会规定特定的解读一阶逻辑的字符串的规则，这些规则就构成了一阶逻辑的语义(sementics)。虽然在不同场合下这些解读并不相同，但它们是具有共性的。这种解读基于自然语言（中文或英文），所以事实上我们假定了自然语言建立在一阶语言之前，称为用来定义一阶语言的元语言(metalanguage)。

Structures

在不同的场合下，同一个一阶逻辑命题可能有不同的含义。例如对于 $\forall v_0 R v_0 v_0$，如果我们是在讨论自然数上的整除关系，那么这个命题会被解读为“任何自然数都能整除自己”，这就是一个“真命题”；而如果我们在讨论实数上的序关系，那么这个命题会被解读为“任何实数都小于自己”，就变成了一个“假命题”。可见，讨论语义时首先需要确定讨论的数学对象。

在上面的例子中，“自然数”与“实数”指明了变量和常量取自哪个集合，这个集合称为论域(universe)，记为 A。选定universe之后，符号集的论域也随之确定。一个 n 元关系就是某个 A^n 的子集，例如当我们说实数 a, b 满足小于关系，就是指有序对 (a, b) 落在 $\mathbb{R}^2$ 的子集 $(a, b) \mid a, b \in \mathbb{R}, a < b$ 当中。同理，一个 n 元函数就是某个 $A^n \to A$ 的映射。每个常数符号对应 A 中某个特定的元素。确定符号集中每个符号的含义，就是确定每个符号对应的元素、集合或映射具体是什么。我们把这个从符号到其具体含义的映射记为 $\mathfrak{a}$，根据定义 $\mathfrak{a}(R) \subseteq A^n$，$\mathfrak{a}(f) : A^n \to A$，$\mathfrak{a}(c_i) \in A$。

A 和 $\mathfrak{a}$ 确定了一阶逻辑中变量的“定义域”和符号集中每个符号的含义。我们把二元组 $(A, \mathfrak{a})$ 记为 $\mathfrak{A}$，称为一个 S 上的结构(S -structure)。为了方便，$\mathfrak{a}(R)$ 常写作 $R^{\mathfrak{A}}$。例如，对于 $S = +, \cdot, <, 0, 1$，取 $A = \mathbb{N}$，$+^{\mathfrak{A}}$ 为自然数的加法运算这一二元函数，$\cdot^{\mathfrak{A}}$ 为自然数的乘法运算这一二元函数，$<^{\mathfrak{A}}$ 为自然数的序关系这一二元关系，$0^{\mathfrak{A}}, 1^{\mathfrak{A}}$ 为加法单位元和乘法单位元这两个常数。这样 $\mathfrak{A} = (\mathbb{N}, \mathfrak{a})$ 就是一个自然数算数的structure，常记为 $\mathfrak{N}^<$，为了方便也可以展开写作 $(\mathbb{N}, +^{\mathbb{N}}, \cdot^{\mathbb{N}}, <^{\mathbb{N}}, 0^{\mathbb{N}}, 1^{\mathbb{N}})$。

Interpretations

Structure给出了变量的定义域和符号集的语义。我们接下来需要定义term和formula的语义。

一个term的语义是从一个term到universe中一个元素的映射。为了确定这个映射，我们首先需要知道term中的每个变量代表universe当中的哪个元素，这就是要我们给出一个 $v_i \to A$ 的映射 β，这个映射称为赋值(assignment)。有了 $\mathfrak{A}$ 和 β，我们就应当可以确定每个term的语义。例如，在structure $\mathfrak{N}^<$ 中令 $\beta(v_0)=1, \beta(v_1)=2$，那么term v_0+v_1 就表示自然数 $1+2$，运算得到 3。所以，我们把一个 S -structure和一个 S -assignment的二元组称为一个 S 下的解释(S -interpretation)，记为 $\mathfrak{I}=(\mathfrak{A},\beta)$。基于结构归纳，定义term的语义为：

- $\mathfrak{I}(v_i)=\beta(v_i)$；
- $\mathfrak{I}(c_i)=c_i^{\mathfrak{A}}$；
- $\mathfrak{I}(ft_1\cdots t_n)=f^{\mathfrak{A}}(\mathfrak{I}(t_1),\cdots,\mathfrak{I}(t_n))$；

formula的语义是从一个term到“真或假”的映射。对于一个formula φ，我们用记号 $\mathfrak{I}\models\varphi$ 来表示 φ 的语义为真，读作interpretation $\mathfrak{I}$ 满足(satisfies) φ，$\models$ 称为满足关系(satisfaction relation)。几个原子性的情况是相当自然的，例如 $\mathfrak{I}\models t_1\equiv t_2$ 当且仅当“ $\mathfrak{I}(t_1)$ 与 $\mathfrak{I}(t_2)$ 是universe当中的同一个元素”为真。关于逻辑连接词的语义也是自然的，例如 $\mathfrak{I}\models\varphi\wedge\psi$ 当且仅当“ $\mathfrak{I}\models\varphi$ 为真并且 $\mathfrak{I}\models\psi$ 为真”为真。唯一需要说明清楚的是自然语言中的“并且、或者”等等究竟表达什么含义，一个清晰的定义方式就是用真值表。复杂的情况是，$\mathfrak{I}\models\forall x\varphi$ 的语义如何定义呢？当变量出现在量词中时，我们实际上并不关心变量的值。当我们说任意元素满足某一性质时，我们并不能认为这是某个特定的元素。$\mathfrak{I}\models\forall x\varphi$ 想表达的其实是，用universe中的每个元素为 φ 中出现的所有 x 的赋值并保持其它的变量的赋值不变，φ 始终为真。为此，我们需要一个描述为formula中的某个特定变量赋特殊的值的方便的符号。对于变量 x,y 和 $a\in A$，定义 $\beta\frac{a}{x}(y):=\begin{cases}\beta(y) & ,y\neq x\\ a & ,y=x\end{cases}$，表示在原assignment中把变量 x 的赋值修改为 a。相应地，$\mathfrak{I}\frac{a}{x}:=(\mathfrak{A},\beta\frac{a}{x})$。这样，$\mathfrak{I}\models\forall x\varphi$ 的语义就可以定义为“对于任意 A 中的元素 a 都有 $\mathfrak{I}\frac{a}{x}\models\varphi$ ”为真。综上，formula的语义定义如下：

- $\mathfrak{I}\models t_1\equiv t_2$ 当且仅当“ $\mathfrak{I}(t_1)$ 与 $\mathfrak{I}(t_2)$ 是universe当中的同一个元素”为真；
- $\mathfrak{I}\models Rt_1\cdots t_n$ 当且仅当“ n 元关系 $R^{\mathfrak{A}}(\mathfrak{I}(t_1),\cdots,\mathfrak{I}(t_n))$ ”为真；
- $\mathfrak{I}\models\neg\varphi$ 当且仅当“ $\mathfrak{I}\models\varphi$ ”为假（也记为 $\mathfrak{I}\not\models\varphi$ ）；
- $\mathfrak{I}\models\varphi\wedge\psi$ 当且仅当“ $\mathfrak{I}\models\varphi$ 为真并且 $\mathfrak{I}\models\psi$ 为真”为真；
- $\mathfrak{I}\models\varphi\wedge\psi$ 当且仅当“ $\mathfrak{I}\models\varphi$ 为真或者 $\mathfrak{I}\models\psi$ 为真”为真；
- $\mathfrak{I}\models\varphi\to\psi$ 当且仅当“ $\mathfrak{I}\models\varphi$ 为假，或者 $\mathfrak{I}\models\psi$ 与 $\mathfrak{I}\models\psi$ 都为真”为真；
- $\mathfrak{I}\models\varphi\leftrightarrow\psi$ 当且仅当“ $\mathfrak{I}\models\varphi$ 与 $\mathfrak{I}\models\psi$ 都为真，或者 $\mathfrak{I}\models\varphi$ 与 $\mathfrak{I}\models\psi$ 都为假”为真；
- $\mathfrak{I}\models\forall x\varphi$ 当且仅当“对于任意 A 中的元素 a 都有 $\mathfrak{I}\frac{a}{x}\models\varphi$ ”为真；
- $\mathfrak{I}\models\exists x\varphi$ 当且仅当“存在 A 中的元素 a 使得 $\mathfrak{I}\frac{a}{x}\models\varphi$ ”为真；

这样，我们就能够用自然语言翻译所有符合一阶逻辑语法的符号串了。

> 一个自然的疑问是，我们为什么要强调structures这一概念的重要性，如果它只是作为引出interpretations这一概念的一个中间步骤？对比二者，相比于structures，interpretations与其唯一的区别在于为变量赋了值。但对于sentences（那些真正刻画本质的formula），对变量的赋值是完全没有意义的，因为所有变量的值都以“存在”或“任意”的方式给出而不是指定。所以对于sentences，我们在讨论语义时给出structure就足够了。此时我们可以把 $\mathfrak{I}\models\varphi$ 简写为 $\mathfrak{A}\models\varphi$ 。

语义上的因果关系

我们已经看到，一个formula可能在有的interpretation下为真，在有的interpretation下为假。而对于同一个符号集 S 下的两个完全不同的interpretation，一个formula可能都为真。甚至，一个formula可能在所有的 S -interpretation下都成立，例如 $\forall x x \equiv x$ ，在任何解释下都最终会被翻译成“universe中的每个元素都和自身相同”。我们把这样的formula称为是语义上恒真的(valid)，记为 $\models \varphi$ ，表示对于任意的 $\mathfrak{I}$ 都有 $\mathfrak{I} \models \varphi$ 。相反，存在命题在任何interpretation下都为假，例如 $\forall x \neg x \equiv x$ 。对于formula φ ，只要存在至少一个 $\mathfrak{I}$ 使得 $\mathfrak{I} \models \varphi$ ，就称 φ 是可满足的(satisfiable)，记为 $\mathrm{Sat}\ \varphi$ 。

更一般地，有的formula之间在语义上有因果关系(consequence relation)。在一个具体数学场合下的因果关系是说如果某个命题A为真那么就能推出命题B为真。这样的因果关系可能在某个interpretation下成立，在另一个interpretation下不成立。但是像例如 $v_0 \equiv v_1 \wedge v_1 \equiv v_2$ 和 $v_0 \equiv v_2$ 这样的命题，在任何解释下如果前者满足就有后者满足，这说明这两个命题本身不依赖于具体解释而具有因果关系。我们称这样的两个formula有语义上的因果关系，仍用符号 $\models$ 表示：如果对于任意的 $\mathfrak{I}$ 都有“如果 $\mathfrak{I} \models \varphi$ 成立就有 $\mathfrak{I} \models \psi$ 成立”为真，就记为 $\varphi \models \psi$ 。

我们可以把符号 $\models$ 的定义推广到formula集合的情况：对于formula集合 Φ ，如果对于所有的 $\varphi \in \Phi$ 都有 $\mathfrak{I} \models \varphi$ ，就记为 $\mathfrak{I} \models \Phi$ 。如果对于所有的 $\varphi \in \Phi$ 都有 $\mathrm{Sat}\ \varphi$ ，就记为 $\mathrm{Sat}\ \Phi$ 。如果对于任何的 $\mathfrak{I}$ 都有“如果 $\mathfrak{I} \models \Phi_1$ 就有 $\mathfrak{I} \models \Phi_2$ ”为真，就记为 $\Phi_1 \models \Phi_2$ 。（由此可见valid的记号 $\models \varphi$ 其实就是 $\varnothing \models \varphi$ 的简写）

语义上的等价关系

对于两个命题 φ, ψ ，如果 $\varphi \models \psi$ 和 $\psi \models \varphi$ 都成立，就称它们是语义上逻辑等价(logically equivalent)的。逻辑等价意味着它们在任何interpretation下的成立都是当且仅当的。

通过简单的结构归纳，我们发现以下等价关系恒成立：

- $\varphi \wedge \psi$ 等价于 $\neg(\neg\varphi \vee \neg\psi)$ ；（我们熟知的De Morgan's Law）
- $\varphi \to \psi$ 等价于 $\neg\varphi \vee \psi$ ；
- $\varphi \leftrightarrow \psi$ 等价于 $\neg(\varphi \vee \psi) \vee \neg(\neg\varphi \vee \neg\psi)$ ；
- $\forall x \varphi$ 等价于 $\neg\exists x \neg\varphi$ ；

这意味着，如果只看表达能力而不看书写的便捷性， $\wedge, \forall, \to, \leftrightarrow$ 本质上是可以被抛弃的，我们只需要 $\neg, \vee, \exists$ 这三个逻辑符号就够了。在做结构归纳中时，对于逻辑符号我们也只需对这三个做归纳即可。

The Coincidence Lemma

对于两个不同的interpretation $\mathfrak{I}_1, \mathfrak{I}_2$ ，什么时候它们对一个term t 的解释是相同的？容易发现，如果 $\mathfrak{I}_1$ 和 $\mathfrak{I}_2$ 本身有相同的universe，且 t 中出现的符号都在其符号集内，两个interpretation对每个变量都有相同的解释，对符号集也有相同的解释， 那么就一定满足 $\mathfrak{I}_1(t) = \mathfrak{I}_2(t)$ 。

同样的，什么时候两个不同的interpretation $\mathfrak{I}_1, \mathfrak{I}_2$ 对一个formula φ 的解释是相同的？自然，对于formula我们只需保证对自由变量（全局变量）有相同解释，同时对符号集有相同解释，就可以保证这一点。

我们把以上两点总结成一条引理，称为The Coincidence Lemma：如果 $\mathfrak{I}_1$ 和 $\mathfrak{I}_2$ 有相同的universe A ，在符号集的交集 $S_1 \cap S_2$ 上对符号都有相同的解释，那么：

- 如果 $\mathfrak{I}_1$ 、 $\mathfrak{I}_2$ 在所有 t 的变量上有相同解释，就成立 $\mathfrak{I}_1(t) = \mathfrak{I}_2(t)$ ；
- 如果 $\mathfrak{I}_1$ 、 $\mathfrak{I}_2$ 在 φ 的所有自由变元上有相同解释，就成立 $\mathfrak{I}_1 \models \varphi \iff \mathfrak{I}_2 \models \varphi$ 。

这一引理可以通过简单的结构归纳严格地说明（用“说明”而不用“证明”一词是为了和之后要讨论的证明的形式化做区分。对这一引理的“证明”是基于自然语言和一阶逻辑语义的定义的，我们认同这一关于语义的性质成立），此处省略。

The Isomorphism Lemma

现在考虑 φ 是sentence，此时 $\mathfrak{I}_1 \models \varphi \iff \mathfrak{I}_2 \models \varphi$ 只需要更弱的条件。首先，对于sentence我们不再需要interpretation的概念，只需退回到structure的概念。其次，我们可以定义structure之间的同构关系(isomorphism)，而不再需要要求universe和符号集的解释完全相同。两个同构的structure本质上是完全相同的，只是元素和符号的名称不同。定义如下：

对于两个 S -structure $\mathfrak{A}$ 和 $\mathfrak{B}$ ，如果它们对应的universe A, B 存在一个 $A \to B$ 的双射 π ，并且对于 S 中的任何 n 元关系符号 R 成立 $(a_1, \cdots, a_n) \in R^{\mathfrak{A}} \iff (\pi(a_1), \cdots, \pi(a_n)) \in R^{\mathfrak{B}}$ ，对于 S 中的任何 n 元函数符号 f 成立 $\pi(f^{\mathfrak{A}}(a_1, \cdots, a_n)) = f^{\mathfrak{B}}(\pi(a_1), \cdots, \pi(a_n))$ ，对于 S 中的任何常数符号 c 成立 $\pi(c^{\mathfrak{A}}) = c^{\mathfrak{B}}$ ，就称 $\mathfrak{A}, \mathfrak{B}$ 是同构的，记为 $\mathfrak{A} \cong \mathfrak{B}$ 。容易证明isomorphism关系是一种等价关系，也即满足自反、对称、传递三条性质。

如果 $\mathfrak{A} \cong \mathfrak{B}$ ，那么对于任意 S -sentence φ ，成立 $\mathfrak{A} \models \varphi \iff \mathfrak{B} \models \varphi$ ，这称为The Isomorphism Lemma。这一引理也可以通过对formula的结构归纳严格说明（注意我们不能对sentence归纳，因为sentence是基于formula定义的对变量有特殊限制的formula，不存在对sentence的归纳定义）。

特别需要指出的一点是，The Isomorphism Lemma的逆命题反向是不对的，也即如果 $\mathfrak{A} \models \varphi \iff \mathfrak{B} \models \varphi$ 对任意 S -sentence φ 成立，并不能推出 $\mathfrak{A}, \mathfrak{B}$ 是isomorphic的。然而这一点在符号集 S 是有限的时候是正确的， S 有限时The Isomorphism Lemma是充分必要的，因为容易想象此时我们可以构造一个sentence，使得满足这个sentence的structure一定具有唯一的结构。然而当 S 是无限集的时候，就无法构造出这样的sentence了（sentence是有限长的），我们之后会看到反例来说明尽管两个structure总是同时满足所有的 S -sentence，但却是不同构的。

The Substitution Lemma

在数学上，我们经常会用到代入(substitution)这一操作：代入操作是指对于某个term t 或formula φ ，将其中的某个变量 x 替换为某个term t' 。

term的代入操作是简单的，应当就是在字符串意义上把所有 x 出现的地方简单地替换为 t' 。对于term t ，把“同时把 t 中的 x_1 代换为 t_1 ， $\cdots$ ， x_n 代换为 t_n ”这一操作记为 $[t]\dfrac{t_1, \cdots, t_r}{x_1, \cdots, x_r}$ ，定义：（这里的中括号和横线并不是一阶逻辑符号，只是一个“记号”，在不会引起歧义的场合可以省略）

- $x\dfrac{t_1, \cdots, t_r}{x_1, \cdots, x_r} := \begin{cases} t_i & , x = x_i \\ x & , \text{otherwise} \end{cases}$;
- $c\dfrac{t_1, \cdots, t_r}{x_1, \cdots, x_r} := c$;
- $[ft'_1 \cdots t'_n]\dfrac{t_1, \cdots, t_r}{x_1, \cdots, x_r} := f[t'_1]\dfrac{t_1, \cdots, t_r}{x_1, \cdots, x_r} \cdots [t'_n]\dfrac{t_1, \cdots, t_r}{x_1, \cdots, x_r}$;

然而我们发现，formula中的代入操作并不应当只是在字符串意义上把所有 x 出现的地方替换为 t 。这是由量词引发的问题，考虑这样一个例子： $\exists z\ z + z \equiv x$ ，在自然数运算的解释 $(\mathfrak{N}, \beta)$ 下这个formula的语义是“ $\beta(x)$ 是偶数”。所以我们期待如果我们把任何变量 y “代入” x ，它的语义都是“ $\beta(y)$ 是偶数”。然而，如果我们把出现在量词中的 z 代入 x ，得到的是 $\exists z\ z + z \equiv z$ ，这个formula的语义显然不是“ $\beta(z)$ 是偶数”而是“存在自然数与自己的和等于自身”，也即“ 0 是自然数”。如果替换新的存在量词， $\exists u\ u + u \equiv z$ ，语义就满足了。所以，我们在代入时应当对量词中的变量做一些特别的关照。对于formula φ ，把“同时把 φ 中的 x_1 代换为 t_1 ， $\cdots$ ， x_n 代换为 t_n ”这一操作记为 $[\varphi]\dfrac{t_1, \cdots, t_r}{x_1, \cdots, x_r}$ ，定义：（有了语义等价，我们可以减少一些归纳的情况了）

- $[t_1' \equiv t_2']\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r} := [t_1']\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r} \equiv [t_2']\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r}$；
- $[Rt_1'\cdots t_n']\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r} := R[t_1']\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r}\cdots[t_n']\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r}$；
- $[\neg\varphi]\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r} := \neg[\varphi]\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r}$；
- $[(\varphi\vee\psi)]\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r} := \left([\varphi]\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r}\vee[\psi]\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r}\right)$；
- 对于 $[\exists x\varphi]\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r}$，取出 $x_1,\cdots,x_r$ 中那些属于 $\text{free}(\exists x\varphi)$ 且 $x_i\neq t_i$ 的变量构成一个子列 $x_{i_1},\cdots,x_{i_s}$。如果 x 不在 $t_{i_1},\cdots,t_{i_s}$ 当中出现，那么 $[\exists x\varphi]\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r} := [\exists x\varphi]\dfrac{t_{i_1},\cdots,t_{i_s}}{x_{i_1},\cdots,x_{i_s}}$；否则，$[\exists x\varphi]\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r} := [\exists u\varphi]\dfrac{t_{i_1},\cdots,t_{i_s},u}{x_{i_1},\cdots,x_{i_s},x}$，其中 u 是不在 $t_{i_1},\cdots,t_{i_s},\varphi$ 中出现的变量，且是 $v_0,v_1,\cdots$ 中下标最小的那个；

我们来理解我们对量词替换的修正。首先，对非自由变量的替换没有意义，把自己替换成自己也没有意义；其次，应当保证新的量词变量不在替换后的项中出现，如果不需要修改量词变量就不修改，否则就修改为从未出现过的一个变量，为了定义的确定性我们规定选择下标最小的那个。

以上替换规则是基于我们想让“替换后语义得以保持”的愿望定义的一系列字符串变换操作。简而言之它会把term中所有想要替换的变量替换成新的项，把formula中的想要替换的自由变量替换成新的项。我们需要验证，这样的变换确实“保持了语义”。而对语义的保持与否体现在用于解释这些term和formula的interpretation中赋值函数 β 是否发生了“合理的变化”。首先，我们推广赋值函数上“替换”的概念：定义 $\beta\dfrac{a_1,\cdots,a_r}{x_1,\cdots,x_r}(y) := \begin{cases}\beta(y) & ,y\neq x_1,\cdots,y\neq x_r\\ a_i & ,y=x_i\end{cases}$，$\mathfrak{I}\dfrac{a_1,\cdots,a_r}{x_1,\cdots,x_r} := \left(\mathfrak{A},\beta\dfrac{a_1,\cdots,a_r}{x_1,\cdots,x_r}\right)$。对于任意interpretation $\mathfrak{I}$，我们想要验证以下两件事成立：

- 对于任意term t，始终成立 $\mathfrak{I}(t\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r}) = \mathfrak{I}\dfrac{\mathfrak{I}(t_1),\cdots,\mathfrak{I}(t_r)}{x_1,\cdots,x_r}(t)$；
- 对于任意formula φ，始终成立 $\mathfrak{I}\models\varphi\dfrac{t_1,\cdots,t_r}{x_1,\cdots,x_r} \iff \mathfrak{I}\dfrac{\mathfrak{I}(t_1),\cdots,\mathfrak{I}(t_r)}{x_1,\cdots,x_r}\models\varphi$。

再一次，我们可以用结构归纳说明以上两点确实成立。这称为The Substitution Lemma。这说明以上定义的“语法”上的替换方案确实达到了我们想在“语义”上达成的目的。