13 集合与子集

Sets and subsets

读书笔记

许博

1 在 λD 中处理子集

之前我们已经在类型理论中，将类型解释为集合，但当考虑到子集时，会有诸多问题需要考虑。比如类型的唯一性与对子集的自然观点矛盾，假设 S 是一个集合而 T 是 S 的真子集，令 $c \in S$，则应有 $c \in T$，也即 $c : S \Rightarrow c : T$，显然违背了类型唯一性。

另一个例子是，令 P 是 S 中的元素的一个属性，集合 $\{x \in S | Px\}$ 表示 S 中满足 P 的所有元素，则对于 $c : S$，$c : \{x \in S | Px\}$ 是不可判定的，因为在类型理论中，类型检查需要是可判定的，即给定的合法的项，一定能够得它的类型匹配与否，而非尚未知晓的答案，且判定任意命题可证明是不可判定的。

类型的可判定性使得类型理论称为用于证明检查与查找的一个强大系统，另一方面，阻止了对于子集的一般处理方式。而类型理论的可判定性以及类型的唯一性需要得到保留，以使我们的推导系统保持作为证明检查基础的有效性。因此，我们决定保留类型理论的可判定性，以及接受这个选择所带来的所有后果，尤其是关于子集的。

在描述子集的表示方式之前，首先介绍数学中与集合有关的一些概念。

$x \in S$ 表示 x 是 S 的一个成员。S 是 T 一个子集，如果 S 的所有成员都是 T 的成员。除此之外还有集合的相等性，并集，交集，差集，补集，幂集以及笛卡尔积等概念，可自行查阅。

在类型理论 λD 中，我们通过一种顺理成章又精彩的方式表示子集：使

用谓词。假设存在一个集合 S，集合 V 是 S 中满足谓词 P 的元素的集合，也即 $V \subseteq S$，显然对于任意的 $x : S$，都有 $x \in V \Leftrightarrow P(x)$。此时，我们使用谓词 P 来表示子集 V，此时对于 S 的子集 V 而言，V 是在 S 上的谓词。如果 $x \in V$，则 Vx 成立。子集的表示如下：

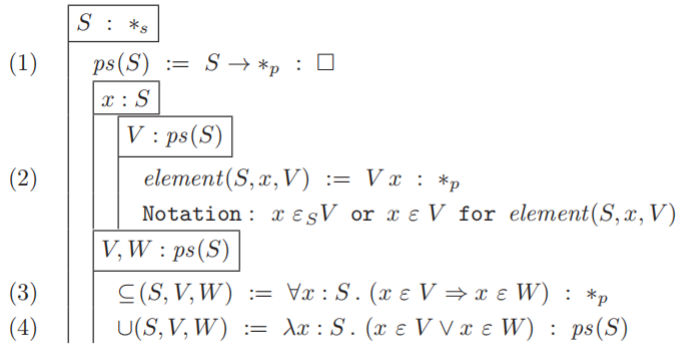

Figure 13.1 Subsets as predicates over a type

其中，$ps(S)$ 表示 $\mathcal{P}(S)$，也即集合 S 的幂集，可见 $ps(S)$ 的成员均是 S 的子集，也即在 S 上的谓词。同时还定义了关系 $\subseteq$ 和 $\cup$。

2 基础集合论概念

本节考虑关于（子）集合的一些概念以及如何在 λD 中形式化它们。关键点在于被形式化为谓词的只有子集。概念 $x \in V$ 等价于 $x \varepsilon V$ 而非 $x : V$，同时 $x \varepsilon V$ 是一个类型。为了确定 $x \varepsilon V$，需要构建一个证明项 $p : x \varepsilon V$。

令 S 是一个解释为集合的类型，V 是 S 的一个子集，V 是在 S 上的一个谓词，所以我们有 $S : *_s$，但是 $V : S \to *_p$。假设我们期望量词作用于子集 V，如 $\forall_{x \in V}(P(x))$，其中 P 是一个谓词，则不能直接形式化为 $\forall x : V.Px$，因为 V 不是一个类型。解决方式是量词作用于集合 S，通过以必须满足谓词 V 的条件以限制定义域，因此使用如下转换：

$\forall_{x \in V}(P(x)) \rightsquigarrow \forall x : S.(x \varepsilon V \Rightarrow Px)$

对于存在量词，使用相似的方式：

$\exists_{x \in V}(P(x)) \rightsquigarrow \exists x : S.(x \varepsilon V \wedge Px)$

可以观察到两者之间分别使用了 $\Rightarrow$ 和 $\rightsquigarrow$，目的是令不满足子集谓词的成员不影响整个命题的真值。

在图 13.2 中，使用子集推导式 $\{x : S | Vx\}$ 表示 $\lambda x : S.Vx$，将 Vx 写

作 $x\varepsilon V$，可以得到 $\{x : S|x\varepsilon V\}$。

以及对于作为类型的集合 S 的子集，和 S 的子集的子集，应用如下变换：

- 若 V 是类型 S 的一个子集，则 $V : ps(S)$，或 $V : S \to *_p$
- 若 V 是 S 的一个子集 W 的一个子集，则 $V \subseteq W$，或 $\forall x : S.(x\varepsilon V \Rightarrow x\varepsilon W)$

在图 13.2 中，定义了子集间的包括和相等，同时定义了子集间的并集，交集，差集，结果同样是子集，以及关于 S 的补集：

$S : *_s$

$V : ps(S)$

Notation : $\{x : S \mid x \varepsilon V\}$ for $\lambda x : S \,.\, V\, x$

$W : ps(S)$

(1) $\subseteq(S, V, W) \;:=\; \forall x : S \,.\, (x \varepsilon V \Rightarrow x \varepsilon W) \;:\; *_p$

Notation : $V \subseteq W$ for $\subseteq(S, V, W)$

(2) $IS(S, V, W) \;:=\; V \subseteq W \wedge W \subseteq V \;:\; *_p$

(3) $\cup(S, V, W) \;:=\; \{x : S \mid x \varepsilon V \vee x \varepsilon W\} \;:\; ps(S)$

(4) $\cap(S, V, W) \;:=\; \{x : S \mid x \varepsilon V \wedge x \varepsilon W\} \;:\; ps(S)$

(5) $\backslash(S, V, W) \;:=\; \{x : S \mid x \varepsilon V \wedge \neg(x \varepsilon W)\} \;:\; ps(S)$

Notation : $V = W,\, V \cup W,\, V \cap W,\, V \backslash W,$ respectively

(6) ${}^c(S, V) \;:=\; \{x : S \mid \neg(x \varepsilon V)\} \;:\; ps(S)$

Notation : V^c for ${}^c(S, V)$

Figure 13.2 Propositions and operations concerning sets

同时非正式地定义了对应操作的中缀形式的语法糖，这些语法糖并未标识出基础的集合 S，而且，我们重载了符号“=”，因为之前定义的“=”用于元素，现在可以用于子集。

考虑表达式 $y\varepsilon\{x : S|Px\}$，等价于 $(\lambda x : S.Px)y$，β-等价于 Py。β-等价可以用于 ε 的引入和消去规则：

$S : *_s$

$P : ps(S)$

$y : S$

$u : P\, y$

(1) $\varepsilon\text{-}in(S, P, y, u) \;:=\; u \;:\; y \,\varepsilon\, \{x : S \mid P\, x\}$

$v : (y \,\varepsilon\, \{x : S \mid P\, x\})$

(2) $\varepsilon\text{-}el(S, P, y, v) \;:=\; v \;:\; P\, y$

Figure 13.3 Introduction and elimination rules for the new element symbol

为了展示这些集合的概念如何工作，给出一个简单的类型论定义的证明，证明定理：

对于 S 的所有子集 V 和 W，如果 $V \subseteq W^c$，则 $V \setminus W = V$。

证明如下：

$S : *_s \mid V, W : ps(S)$

$x : S$

$v : (x \varepsilon V \backslash W)$

(1) $a_1^\dagger := v$ (or : use ε-el) : $x \varepsilon V \wedge \neg(x \varepsilon W)$

(2) $a_2 :=$ ... use $\wedge$-el_1 on a_1 ... : $x \varepsilon V$

(3) $a_3 :=$ ... use $\Rightarrow$-in on a_2 ... : $x \varepsilon V \backslash W \Rightarrow x \varepsilon V$

(4) $a_4(S, V, W) :=$ ... use $\forall$-in on a_3 ... : $V \backslash W \subseteq V$

$u : V \subseteq W^c$

$x : S$

$v : x \varepsilon V$

(5) $a_5 := u\, x\, v : x \varepsilon W^c$

(6) $a_6 := a_5 : \neg(x \varepsilon W)$

(7) $a_7 :=$ ... use $\wedge$-in on v and a_6 ... : $x \varepsilon V \wedge \neg(x \varepsilon W)$

(8) $a_8 := a_7 : x \varepsilon V \backslash W$

(9) $a_9 :=$ ... use $\Rightarrow$-in on a_8 ... : $(x \varepsilon V) \Rightarrow (x \varepsilon V \backslash W)$

(10) $a_{10}(S, V, W, u) :=$ ... use $\forall$-in on a_9 ... : $V \subseteq V \backslash W$

(11) $a_{11}(S, V, W, u) :=$ ... use $\wedge$-in on $a_4(S, V, W)$ and $a_{10}(S, V, W, u)$ ... : $V \backslash W = V$

(12) $a_{12}(S, V, W) :=$ ... use $\Rightarrow$-in on a_{11} ... : $(V \subseteq W^c) \Rightarrow (V \backslash W = V)$

†*parameters suppressed*

需要注意的是，在证明过程中省略了一些基于逻辑规则的证明项，使用了注解的形式而非正式的使用标准形式的定义实例化，以提高可读性。

本节中所涉及到的概念还剩下一个问题：我们所定义的子集的等价 $IS(S, V, W)$，与在章节 12.2 中讨论的莱布尼兹等价如何联系？

首先我们有 $S : *_p$，但是 $ps(S) : \square$，因此不能使用 $V =_{ps(S)} W$ 以表示 V 和 W 的莱布尼兹等价，但是可以定义相似的在 S 的幂集上的莱布尼兹等价：$\Pi K : ps(S) \to *_p.(KV \Leftrightarrow KW)$，使用 $V \triangleq_{ps(S)} W$ 表示。

莱布尼兹等价易于证明子集等价，但翻转之后不能构建出我们需要的证明。因此需要添加另一个公理：

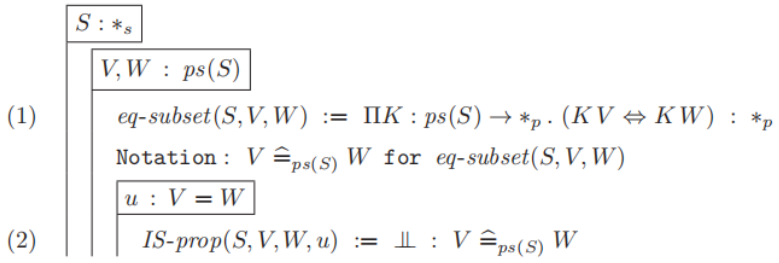

Figure 13.5 $V = W$ implies $V \widehat{=}_{ps(S)} W$

3 特殊的子集

另一个关于集合的基础概念是空集，不包含任何元素的集合。在我们使用幂集作为子集的类型时，我们不能获得一个通用的空集，必须定义对于每个类型 $S : *_s$，有一个关于给定 S 的空集 $\emptyset(S)$。这样的一个 $\emptyset(S)$（表示为 $\emptyset_S$）是 S 的一个子集，编码为 S 上的谓词 $\lambda x : S. \bot$。

同样定义 S 的全子集，包含 S 中的所有元素，编码为谓词 $\lambda x : S.\neg \bot$：

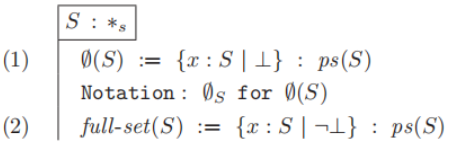

Figure 13.6 The empty set and full set as subsets of a type S

图 13.8 中，证明对于一个子集，如果它不是空集，则至少包含一个元素：

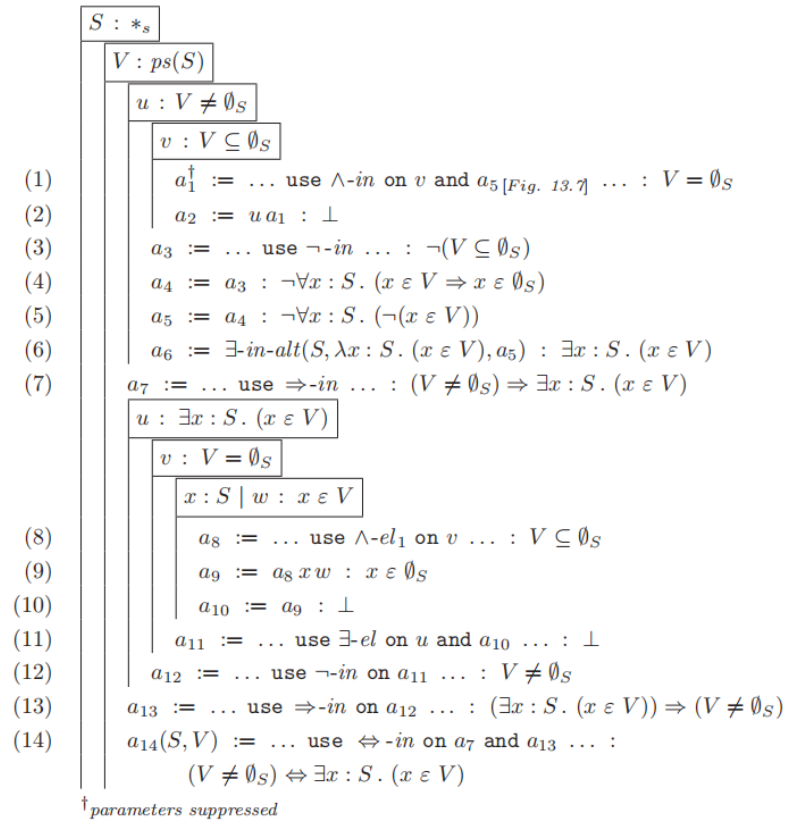

Figure 13.8 Properties of the empty set

4 关系

在已经知道如何在 λD 中表示集合后，将了解与集合有关的概念。首先是关系。

一个关系在 S 上的关系是一个在 S 上的二元谓词，具有类型 $S \to S \to *_p$。在之前提到过，关系可能具有一些具体的性质，比如自反性，对称性或者传递性。一个具有这三个性质的关系被称作等价关系。这些关系在 λD 中易于表示：

$S : *_s$

$R : S \to S \to *_p$

(1) $\quad reflexive(S, R) := \forall x : S\,.\,(R\,x\,x) : *_p$

(2) $\quad symmetric(S, R) := \forall x, y : S\,.\,(R\,x\,y \Rightarrow R\,y\,x) : *_p$

(3) $\quad transitive(S, R) := \forall x, y, z : S\,.\,(R\,x\,y \Rightarrow R\,y\,z \Rightarrow R\,x\,z) : *_p$

(4) $\quad equivalence\text{-}relation(S, R) :=$

$\quad\quad reflexive(S, R) \wedge symmetric(S, R) \wedge transitive(S, R) : *_p$

Figure 13.9 Basic notions connected to relations

同样可以定义在子集上的关系的性质，比如：

$refl - subset(S, V, R) := \forall x : S.(x \varepsilon V \Rightarrow Rxx)$。

一个等价关系可以将集合切分为非空的若干等价类：x 的等价类包含所有与 x 具有该等价关系的 y。在图 13.10 中定义，使用 $[x]_R$ 简写表示 $class(S, R, u, x)$，省略两个参数：

$S : *_s$

$R : S \to S \to *_p$

$u : equivalence\text{-}relation(S, R)$

$x : S$

$class(S, R, u, x) := \{y : S \mid R\,x\,y\} : ps(S)$

`Notation` : $[x]_R$ `for` $class(S, R, u, x)$

Figure 13.10 Equivalence classes

等价类具有三个重要性质：

(1) $\forall x : S\,.\,([x]_R \neq \emptyset_S)$,

(2) $\forall x, y : S\,.\,([x]_R = [y]_R \vee ([x]_R \cap [y]_R = \emptyset_S))$,

(3) $\forall y : S\,.\,\exists x : S\,.\,(y\ \varepsilon\ [x]_R)$.

证明与 (2) 等价的命题：$\forall x, y, z : S.(z \varepsilon\, [x]_R \Rightarrow z \varepsilon\, [y]_R \Rightarrow ([x]_R = [y]_R))$：

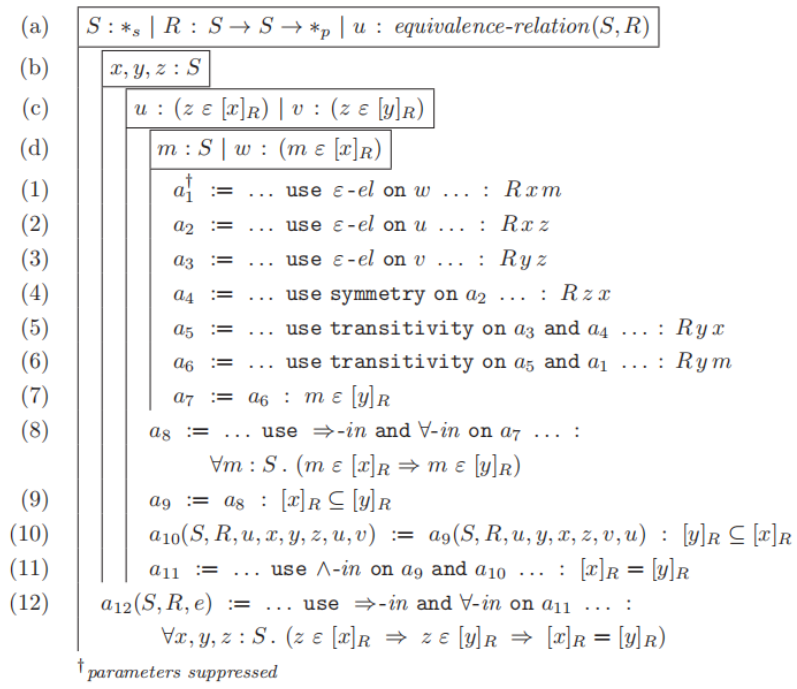

Figure 13.11 Proof of a lemma about equivalence classes

至此，我们已经讨论了在一个类型 S 上的关系 R，一个自然的扩展是在一对类型（比如 S 和 T）上的关系，这样的一个关系的显然表示具有类型 $S \to T \to *_p$。

5 映射

映射可以看作是一类特殊的关系，更确切地说：从集合 S 到集合 T 的映射是一个关系 $F : S \to T \to *_p$，即：

$\forall_{x \in S} \exists^1_{y \in T} (Fxy)$

所以将一个关系变成映射的关键性质为每一个 $x \in S$ 都和一个确定的 $y \in T$ 具有一个关系。称这样的一个关系为函数式关系。由于与 x 关系的 y 的唯一性，通常将 F 作为一个一元符号以及写作 $F(x) = y$ 而非 Fxy。

作为一个 Π-类型的成员的映射与作为一个函数式关系的映射之间的关联很容易确定：

$S,T : *_s$

$F : S \to T$

(1) $R(S,T,F) \;:=\; \lambda x : S\,.\,\lambda y : T\,.\,(y =_T F\,x) \;:\; S \to T \to *_p$

(2) $a_2(S,T,F) \;:=\; \ldots\ \texttt{Exerc. 13.12 (a)}\ldots \;:$
$\forall x : S\,.\,\exists^1 y : T\,.\,(R(S,T,F)\,x\,y)$

$R : S \to T \to *_p$

$u \;:\; \forall x : S\,.\,\exists^1 y : T\,.\,R\,x\,y$

(3) $F(S,T,R,u) \;:=\; \lambda x : S\,.\,\iota^{ux}_{y:T}(R\,x\,y) \;:\; S \to T$

Figure 13.12 The connection between a functional relation and a type-theoretic function

即，由一个抽象可以得到一个函数式关系，而由一个函数式关系，也可以得到一个抽象。前者称作是类型理论格式的映射，而我们将继续使用类型理论格式表示，因为它在我们的框架中更为基础。

映射 $F : S \to T$ 可能具有诸如内射性，满射性以及双射性。同样可以直接表达的概念式一个满足双射性映射的逆：

$S,T : *_s$

$F : S \to T$

(1) $injective(S,T,F) \;:=\; \forall x_1, x_2 : S\,.\,(F\,x_1 =_T F\,x_2 \Rightarrow x_1 =_S x_2) \;:\; *_p$

(2) $surjective(S,T,F) \;:=\; \forall y : T\,.\,\exists x : S\,.\,(F\,x =_T y) \;:\; *_p$

(3) $bijective(S,T,F) \;:=\; injective(S,T,F) \wedge surjective(S,T,F) \;:\; *_p$

$u : bijective(S,T,F)$

(4) $a_4(S,T,F,u) \;:=\; \ldots\ \texttt{Exerc. 13.12 (b)}\ldots \;:$
$\forall y : T\,.\,\exists^1 x : S\,.\,(F\,x =_T y)$

(5) $inv(S,T,F,u) \;:=\; \lambda y : T\,.\,\iota^{a_4(S,T,F,u)\,y}_{x:S}(F\,x =_T y) \;:\; T \to S$

Figure 13.13 Some well-known notions connected with maps

当函数 F 的域非类型 S 而是子类时，情况会变得有一些复杂，如：

$S,T : *_s$

$V : ps(S)$

$F \;:\; \Pi x : S\,.\,((x\ \varepsilon\ V) \to T)$

(1) $inj\text{-}subset(S,T,V,F) \;:=\; \forall x_1, x_2 : S\,.\,\Pi p : (x_1\ \varepsilon\ V)\,.\,\Pi q : (x_2\ \varepsilon\ V)\,.$
$((F\,x_1\,p =_T F\,x_2\,q) \Rightarrow x_1 =_S x_2) \;:\; *_p$

Figure 13.14 Injectivity of a map on a subset

其它概念同样易于形式化，给定集合 S 和 T，以及一个函数 $F : S \to T$：

- 源集合 S 的一个子集 V 的 F-像；V 的像是 T 的一个子集，包括 V 中元素对用的所有 F-值。
- 范围集合 T 的一个子集 W 的 F-原像；W 的原像是 S 的子集，包括所有在 W 中具有 F-值的元素。

$$
\begin{array}{|l}
\hline
S,T : *_s \\
\quad \boxed{F : S \to T} \\
\quad\quad \boxed{V : ps(S)} \\
(1) \quad\quad\quad image(S,T,F,V) \;:=\; \{y : T \mid \exists x : S \,.\, (x \;\varepsilon\; V \wedge F\,x =_T y)\} \;:\; ps(T) \\
\quad\quad \boxed{W : ps(T)} \\
(2) \quad\quad\quad origin(S,T,F,W) \;:=\; \{x : S \mid F\,x \;\varepsilon\; W\} \;:\; ps(S) \\
\hline
\end{array}
$$

Figure 13.15 Image and origin of a subset

6 数学概念的表示

在图 13.17 中，给出了一些基本概念的列表，这些基本概念在数学中具有不同的含义，但却以相同的 λD-构造表示：

mathematics	**the type theory of λD**	
function space $A \to B$	$\Pi x : A \,.\, B$	$A : *_s,\ B : *_s$
implication $A \Rightarrow B$	$\Pi x : A \,.\, B$	$A : *_p,\ B : *_p$
universal statement $\forall_{x \in A}(B(x))$	$\Pi x : A \,.\, B$	$A : *_s,\ B : *_p$
function from A to B	$\lambda x : A \,.\, t$	$A : *_s,\ t : B : *_s$
predicate on A	$\lambda x : A \,.\, t$	$A : *_s,\ t : *_p$
subset of A	$\lambda x : A \,.\, t$	$A : *_s,\ t : *_p$
two-valued function from $A \times B$ to C	$\lambda x : A \,.\, \lambda y : B \,.\, t$	$A, B : *_s,\ t : C : *_s$
binary relation R over $A \times B$	$\lambda x : A \,.\, \lambda y : B \,.\, t$	$A, B : *_s,\ t : *_p$

Figure 13.17 Coding mathematics in λD

优点在于，令 λD 系统保持相对简单，以及数学和逻辑中相似的计算模式在 λD 中重合。另外，一个显然的缺点是对于 λD 表达式的翻译会有歧义，当不给出额外的说明时。

但当我们使用 $*_s$ 以及 $*p$ 时，这个缺点会几乎不见，唯一的例外在于类型上的谓词和对应类型的子集之间无法区分，这也是在章节 13.1 中我们选择承担维护系统可判定行的后果。