计算机科学 (Haskell) 中的偏序理论（译，上）

2018, Jan 2 by Tesla Ice Zhang

这是**我第一篇看懂**的介绍偏序理论的文章（也可能是因为我之前看一些中文资料的时候没学过 Agda）。

原文地址

下面是翻译内容。

每一个字都是我手打的，名词的翻译参考了 Wikipedia (在最后会列出参考)，没有一丝一毫的机翻和复制。

由于原文篇幅比较长，因此我将分为两部分翻译。

计算机科学中的偏序理论

偏序理论(partial order theory)在计算机科学里面有很广泛的应用，尤其是逻辑、形式方法、程序语言和静态分析方面。

但是！因为偏序理论和其他领域（比如拓扑学）绑在一起，涉及一大堆定义和各种概念，不是所有的这些都经常在计算机科学里面被使用。

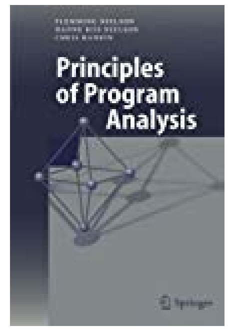
所以说，在这篇文章中，我总结了偏序理论在计算机科学中用的最多的那部分——尤其是在静态分析中被使用的那部分。

本文话题包含前序(preorders)、等价关系(equivalence relations)、全序(total orders)、偏序(partial orders)、半格

(semilattices)、格(lattices)、有界格(bounded lattices)和完备格(complete lattices)、 Scott 连续性(Scott-continuity)、单调性、不动点和 Kleene 不动点定理(Kleene's fixed point theorem)。

在本文最后，你会看到偏序与格在幂集上的自然升格(natural liftings)、笛卡尔积、序列(sequences)和函数空间。

形式定义会在可能的情况下会给出对应的 Haskell 代码表示。

【如果你对偏序理论在静态分析里的应用很感兴趣，你应该看看这本《Principles of Program Analysis》的附录：

】

Haskell 代码

在本文所使用的 Haskell 代码文件的开头是这样的（吐槽：这是在教人如何把 Haskell 写成 Agda 吗。。。。）：

```
import Data.Map as Map hiding (map)
import Data.Set as Set hiding (map)

type ℙ a = Set.Set a

type k :-> v = Map.Map k v
```

然后我们还需要开这些 GHC 扩展：

```
-XTypeOperators -XTypeSynonymInstances -XParallelList
```

前序

如果二元关系 R 满足 $\forall x. x \ R \ x$ ，我们可以说 R 满足**自反性**(reflexive)。

如果二元关系 R 满足 $a \ R \ b$ 且 $b \ R \ c$ 蕴含 $a \ R \ c$，我们可以说 R 满足**传递性**(transitive)。

当且仅当集合 X 上的二元关系 $\sqsubseteq$ 同时满足自反性和传递性，我们称 $\sqsubseteq$ 为一个**前序**。

相等关系

如果二元关系 R 满足 $a \ R \ b$ 蕴含 $b \ R \ a$，我们可以说 R 是**对称**(symmetric)的。

一个**相等关系** $\equiv$ 是一个对称(symmetric)的前序。

一个相等关系 $\equiv$ 将集合 S 分为不同的**等价类**(equivalence classes)。

元素 x 的等价类是 $[x]_{\equiv}$ ：

$$[x]_{\equiv} \stackrel{def}{=} \{y | x \equiv y\}$$

由所有的 X 上的 $\equiv$ 关系下的等价类构成的集合是 $X \ / \equiv$ 。

由前序得来的等价关系

所有的前序 $\sqsubseteq$ 包含了一个自然等价关系 $\equiv_{\sqsubseteq}$ ：

$$a \equiv_{\sqsubseteq} b \text{ iff } a \sqsubseteq b \text{ and } b \sqsubseteq a.$$

偏序

如果二元关系 R 满足 $a \ R \ b$ 和 $b \ R \ a$ 蕴含 $a = b$，我们可以说 R 是**反对称**(antisymmetric)的。

反对称的前序被称为**偏序**。

具有偏序 $\sqsubseteq$ 集合 X 被称为**偏序集**(partial order set / poset)，一般被表示为 $(X, \sqsubseteq)$。

在 Haskell 中，我们可以将偏序集定义为一个 typeclass 。

```
class PartialOrder t where
 (⊑) :: t -> t -> Bool
```

全序

如果二元关系 R 满足 $\forall a\ b. a\ R\ b$ 或 $b\ R\ a$，那么我们可以说 R 是**完全**(total)的。

如果二元关系 $\leq$ 反对称、完全，并具有传递性，那么我们称 $\leq$ 为一个**全序**。

完全的性质蕴含了自反性，也就是说所有的全序都是偏序。

交半格

a 和 b 的**最大下界**(greatest lower bound)为同时小于它们俩的最大元素，记作 $a \sqcap b$ 。

如果一个偏序集 X 里任意两个元素都有最大下界，那么我们可以说 X 是一个**交半格**(meet semilattice)。

在格中，最大下界必须是唯一的。

a 和 b 的最大下界也被称为 a 和 b 的交，或者下确界。

最大下界有这些性质：

- $(a \sqcap b) \sqsubseteq a$
- $(a \sqcap b) \sqsubseteq b$
- $c \sqsubseteq a$ 与 $c \sqsubseteq b$ 蕴含 $c \sqsubseteq (a \sqcap b)$

并半格

a 和 b 的**最小上界**(least upper bound)为同时大于它们俩的最小元素，记作 $a \sqcup b$ 。

如果一个偏序集 X 里任意两个元素都有最小上界，那么我们可以说 X 是一个**并半格**(join semilattice)。

在格中，最小上界必须是唯一的。

a 和 b 的最小上界也被称为 a 和 b 的并，或者上确界。

最小上界有这些性质：

- $a \sqsubseteq (a \sqcup b)$
- $b \sqsubseteq (a \sqcup b)$
- $a \sqsubseteq c$ 与 $b \sqsubseteq c$ 蕴含 $(a \sqcup b) \sqsubseteq c$

格

如果一个偏序集既是交半格又是并半格，那么我们可以说这个偏序集是一个**格**。

在 Haskell 中，我们可以将格定义为一个具有交和并的偏序集：

```
class PartialOrder t => Lattice t where
  (⊔) :: t -> t -> t
  (⊓) :: t -> t -> t
```

有界格

如果一个格 $(L, \sqsubseteq)$ 同时具有 L 的最大元素(顶，或者 $\top$)和最小元素(底，或者 $\bot$)，那么我们可以说它是**有界**(bounded)的。

对于所有有界格中的元素 x ，都一定有

- $x \sqsubseteq \top$ 且 $\bot \sqsubseteq x$

在 Haskell 中，我们可以将有界格定义为一个具有顶和底的格：

```
class Lattice t => BoundedLattice t where
  bot :: t
  top :: t
```

完备格

如果一个格 $(L, \sqsubseteq)$ 的 L 的所有（有可能有无穷多个）子集 S 都同时具有最大下界($sup(S)$)和最小上界($inf(S)$)，那么我们可以称之为一个**完备格**。
所有完备格 $(L, \sqsubseteq)$ 都是有界格：

- $\bot = inf(L)$
- $\top = sup(L)$

单调函数

给定偏序集 $(X, \sqsubseteq_X)$ 和 $(Y, \sqsubseteq_Y)$，如果函数 $f : X \to Y$ 存在 “ $x \sqsubseteq_X x'$ 蕴含 $f(x) \sqsubseteq_Y f(x')$ “ 这样的性质，那么我们可以说函数 f 是**单调**(monotonic)的。

连续函数

为了定义连续函数，我们需要先定义函数在集合上的按成员应用(member-wise function application across sets)。
对于函数 $f : X \to Y$ ，如果 $S \subseteq X$ ，那么
$f. S = \{f(x) \mid x \subseteq S\}$。

相应的，也可以说 $f.\{x_1,\ldots,x_n\} \stackrel{def}{=} \{f(x_1),\ldots,f(x_n)\}$ 。（在一些文本中，函数在集合上的按成员应用和普通的函数应用并没有在符号上被区分出来。）

给定格 $(X,\sqsubseteq_X)$ 和 $(Y,\sqsubseteq_Y)$ ，如果函数 $f : X \rightarrow Y$ 存在 “ $S \subseteq X$ 蕴含 $f(sup(S)) = sup(f.S)$ “ 这样的性质，那么我们可以说函数 f 是**Scott 连续**(Scott-continuous)的。

Scott 连续的函数都是单调的。

不动点

给定函数 $f : X \rightarrow X$， 如果 $x = f(x)$ ，那么我们说 x 是 f 的一个**不动点**。

区域

对于完备格 $(X,\sqsubseteq)$ 上的函数 $f : X \rightarrow X$，我们可以把集合 X 分为一些区域(region)：

- $Fix(f) = \{x \mid x = f(x)\}$ 是一个不动点区域。
- $Asc(f) = \{x \mid x \sqsubseteq f(x)\}$ 是一个上升区域。
- $Desc(f) = \{x \mid x \sqsupseteq f(x)\}$ 是一个下降区域。

我们还应该区分最大不动点和最小不动点：

- $lfp(f) = inf(Fix(f))$
- $gfp(f) = sup(Fix(f))$

这些区域有以下性质：

- $Fix(f) = Asc(f) \cap Desc(f)$
- 如果 $x \in Asc(f)$，那么 $f(x) \in Asc(f)$
- 如果 $x \in Desc(f)$，那么 $f(x) \in Desc(f)$
- $\bot \in Asc(f)$
- $\top \in Desc(f)$

我建议将上面的性质的证明留作习题，答案略。读者自证不难。

Kleene 链

给定格 $(L, \sqsubseteq)$ 上的单调函数 $f : L \to L$，我们称集合 $K(x)$ 为从 $x \in L$ 开始的 **Kleene 链**(Kleene chain)：

$$K(x) \stackrel{def}{=} \{f^i(x) \mid i \geq 0\}$$

符号 f^i 表示函数的 i 次复合：

- $f^0(x) = x$
- $f^i(x) = f^{i-1}(f(x))$

如果 $x \in Asc(f)$，那么会有一个对链 $K(x)$ 的上升序列(ascending order)，因为 $\forall x' \in Asc(f)\ .\ f^i(x') \sqsubseteq f^{i+1}$ 。在 Haskell 中，一个无限的 `List` 就表示一个从 $\bot$ (底)开始的 Kleene 链。

```
kleene :: (BoundedLattice t) => (t -> t) -> [t]
kleene f = bot : (f <$> kleene f)
```

Kleene 不动点定理

在格上，**Kleene 不动点定理**(Kleene’s fixed point theorem)的内容是：

> 如果 $(L, \sqsubseteq)$ 是一个完备格，并且函数 $f : L \rightarrow L$ 是连续的，那么有：
> $lfp(f) = sup(K(\bot))$

更多地，对于一个有限高度(height)的格，总有一个自然数 n 满足：

$$lfp(f) = f^n(\bot)$$

这一事实带来了一个计算不动点的简单算法。
这个 `stable` 函数首先拿出一个序列中的第一个元素，然后重复它后面的元素。

```
stable :: Eq a => [a] -> a
stable (x : fx : tl)
  | x == fx   = x
  | otherwise = stable $ fx : tl
```

所以说，这个最小的不动点(least fixed point, lfp)函数就是从一个 Kleene 序列中找这个 `stable` 的点的。

```
lfp :: (BoundedLattice t, Eq t) => (t -> t) -> t
lfp = stable . kleene
```

未完待续

但是这个”上”我就已经写了五天了。。。

新年快乐。

Tweet this

Top

创建一个 issue 以申请评论

Create an issue to apply for commentary

协议/License

| |