验证脚本与论文关系说明（Question and Answer）

问题 1：论文中 BsTreeVar 对应脚本中的位置在哪？

在论文的 2.1 节，我们给出了二叉搜索树变体(简称 BsTreeVar)的抽象结构定义。BsTreeVar 是带附加属性的二叉搜索树抽象类型，基于该类型给出了二叉搜索树类算法的函数式建模框架和验证框架。在本文工作中，以 LLRB 的建模和相关操作的正确性证明作为实际问题，以检验抽象类型的建模框架和验证框架的有效性。

由于 BsTreeVar 是抽象的，在对具体的数据结构 LLRB 进行建模时，需要进行实例化，文件 LLRB_SET.thy 第 9-10 行给出了 LLRB 的结构定义，其中 BsTreeVar 实例化为 llrb，add_prop 实例化为 color：

type_synonym 'a **BsTreeVar** = ('a × **add_prop**) search_tree

```
datatype color = Red | Black
type_synonym 'a llrb = "('a*color) search_tree"
```

图 1-1　文件 LLRB_SET.thy 中的第 9-10 行

问题 2：论文图 1 中所提及的 5 个公式对应脚本的位置在哪？

关于论文中的图 1 所提及的 5 个公式可以在文件 Set_Specs.thy 中找到，如图 1-2 所示：

Set_Specs.thy (F:\hardworking\lab\laboratory\thesis\llrb\LLRB_PROOF_12.4\)

```
locale Set =
fixes empty :: "'s"
fixes insert :: "'a ⇒ 's ⇒ 's"
fixes delete :: "'a ⇒ 's ⇒ 's"
fixes lookup :: "'s ⇒ 'a ⇒ bool"
fixes set_search_tree :: "'s ⇒ 'a set"
fixes invar :: "'s ⇒ bool"
assumes set_empty:    "set_search_tree empty = {}"
assumes set_insert:   "invar BsTreeVar ⟹ set_search_tree(insert x BsTreeVar) = {x} ∪ set_search_tree BsTreeV
assumes set_delete:   "invar BsTreeVar ⟹ set_search_tree(delete x BsTreeVar) = set_search_tree BsTreeVar - {
assumes set_lookup:   "invar BsTreeVar ⟹ lookup BsTreeVar x = (x ∈ set_search_tree BsTreeVar)"
assumes invar_empty:  "invar empty"
assumes invar_insert: "invar BsTreeVar ⟹ invar(insert x BsTreeVar)"
assumes invar_delete: "invar BsTreeVar ⟹ invar(delete x BsTreeVar)"
```

图 1-2　文件 Set_Specs.thy 中的第 17-22 行

问题 3：论文第 4 页所提及的 locale BsTreeVar_op 在脚本中的位置在哪？

与问题 1 类似，BsTreeVar_op 是 BsTreeVar 抽象类型的相关操作。本文以 LLRB 的建模和相关操作的正确性证明作为实际问题，以检验抽象类型的建模框架和验证框架的有效性。本文针对 LLRB，给出了具体类型 llrb 的相关操作 llrb_op，如

图 1-3 所示，BsTreeVar_op 实例化为 llrb_op，BsTreeVar 实例化为 llrb。。

我们在论文的 2.3 节中给出了 locale BsTreeVar_op 的定义：

$$
\begin{aligned}
&locale\ \ BsTreeVar\_op\ =\\
&\quad fixes\ \ pre\_inv_l ::"'a\ \ BsTreeVar \Rightarrow 'a\ \ BsTreeVar \Rightarrow 'a \Rightarrow 'a\ \ BsTreeVar \Rightarrow 'a\ \ BsTreeVar"\\
&\quad and\ \ \ pre\_inv_r ::"'a\ \ BsTreeVar \Rightarrow 'a \Rightarrow 'a\ \ BsTreeVar \Rightarrow 'a\ \ BsTreeVar \Rightarrow 'a\ \ BsTreeVar"\\
&\quad and\ \ \ pre\_inv_{split} ::"'a\ \ BsTreeVar \Rightarrow 'a \Rightarrow 'a\ \ BsTreeVar \Rightarrow 'a\ \ Bs\ TreeVar"
\end{aligned}
$$

proof_delete.thy (%USERPROFILE%\Desktop\LLRB_PROOF_12.4\)

```
subsection ‹locale implementation›

locale llrb_op =
fixes pre_invl::"('a::linorder) llrb ⇒'a llrb ⇒'a ⇒ 'a llrb  ⇒ 'a llrb"
  and pre_invr::"'a llrb ⇒ 'a ⇒ 'a llrb  ⇒'a llrb ⇒ 'a llrb"
  and pre_invsplit::"'a llrb ⇒'a ⇒ 'a llrb  ⇒ 'a llrb"
```

图 1-3　文件 proof_delete.thy 中的第 47-50 行

***********以下补充说明了区域 llrb_op 中所定义的泛化函数 ins'和 del'、泛化函数的实例化过程以及实例化函数的正确性验证。***********

① **泛化函数 ins'和 del'**

基于论文 2.3 节的二叉搜索树类结构的插入操作泛化函数 ins'，我们把 BsTreeVar 实例化为 llrb，从而得到 LLRB 的插入操作的泛化函数 ins'，如图 1-4 所示。

$$
\begin{aligned}
&fun\ ins'::"'a \Rightarrow 'a\ \ BsTreeVar \Rightarrow 'a\ \ BsTreeVar"\\
&\quad ins'\ x\ <>\ =\ <>,(x,\_),<>>\\
&\quad ins'\ x\ <l,(a,\_),r>\ =\ (case\ cmp\ x\ a\ of\ \ LT \Rightarrow pre\_inv_l\ (ins\ x\ l)\ l\ a\ r\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad GT \Rightarrow pre\_inv_r\ \ l\ a\ r\ \ (ins\ x\ r)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad EQ \Rightarrow l\ a\ r)
\end{aligned}
$$

proof_delete.thy (%USERPROFILE%\Desktop\LLRB_PROOF_12.4\)

```
fun ins' :: "'a::linorder ⇒ 'a llrb ⇒ 'a llrb" where
  "ins' x Leaf = R Leaf x Leaf" |
  "ins' x (B l a r) = (case cmp x a of
            LT ⇒ pre_invl (ins' x l) l a r |
            GT ⇒ pre_invr l a r (ins' x r) |
            EQ ⇒ B l a r)" |
  "ins' x (R l a r) = (case cmp x a of
            LT ⇒ R (ins x l) a r |
            GT ⇒ R l a (ins x r) |
            EQ ⇒ R l a r)"
```

图 1-4　文件 proof_delete.thy 中的第 60-69 行

基于论文 2.3 节的二叉搜索树类结构的删除操作泛化函数 del'，我们把 BsTreeVar 实例化为 llrb，从而得到 LLRB 的删除操作的泛化函数 del'，如图 1-5 所示。

$$fun\ del'::"'a \Rightarrow 'a\ BsTreeVar \Rightarrow 'a\ BsTreeVar"$$
$$del'\ x <> = <>$$
$$del'\ x < l,(a,\_),r >= (case\ cmp\ x\ a\ of\ LT \Rightarrow pre\_inv_l\ (del\ x\ l)\ l\ a\ r$$
$$GT \Rightarrow pre\_inv_r\ l\ a\ r\ (del\ x\ r)$$
$$EQ \Rightarrow pre\_inv_{split}\ l\ a\ r)$$

```
fun del'::"'a::linorder  ⇒ 'a llrb ⇒ 'a llrb" where
   "del' x Leaf = Leaf"|
   "del' x (Node l (a, _) r) = (case cmp x a of
              LT ⇒ pre_invl (del' x l) l a r|
              GT ⇒ pre_invr l a r (del' x r)|
              EQ ⇒ pre_invsplit l a r)"
```

图 1-5　文件 proof_delete.thy 中的第 53-58 行

② 泛化函数的实例化过程

解释区域 llrb_op，将图 1-3 中的接口 pre_invl 和 pre_invr 分别实例化为函数 baliL'和 baliR'，从而将 LLRB 插入函数 ins'实例化，如图 1-6 所示。

```
proof_delete.thy (F:\hardworking\lab\laboratory\thesis\llrb\LLRB_PROOF_12.4\)
locale llrb_insert
begin
fun baliL' :: "('a ::linorder) llrb ⇒'a llrb ⇒ 'a ⇒ 'a llrb ⇒ 'a llrb" where
"baliL' (R t1 a (R t2 b t3)) _ c t4 = R (B t1 a t2) b (rightredB t3 c t4)" |
"baliL' (R (R t1 a t2) b t3) _ c t4 = R (B t1 a t2) b (rightredB t3 c t4)" |
"baliL' t1 _ a t2 = rightredB t1 a t2"

fun baliR' :: "('a ::linorder) llrb ⇒ 'a ⇒ 'a llrb ⇒ 'a llrb ⇒'a llrb" where
"baliR' t1 a _ (R t2 b (R t3 c t4)) = R (rightredB t1 a t2) b (B t3 c t4)" |
"baliR' t1 a _ (R (R t2 b t3) c t4) = R (B t1 a t2) b (rightredB t3 c t4)" |
"baliR' t1 a _ t2 = rightredB t1 a t2"

interpretation llrb_op baliL' baliR'
done
```

图 1-6　文件 proof_delete.thy 中的第 103-116 行

解释区域 llrb_op，将图 1-3 中的接口 pre_invl、pre_invr 和 pre_invsplit 分别实例化为函数 pre_invl'、pre_invr'和 pre_invsplit'，从而将 LLRB 删除函数 del'实例化，如图 1-7 所示。

```
proof_delete.thy (%USERPROFILE%\Desktop\LLRB_PROOF2\)
locale llrb_delete
begin
definition pre_invl'::"('a::linorder) llrb ⇒'a llrb ⇒ 'a ⇒ 'a llrb ⇒ 'a llrb "

definition pre_invr'::"('a ::linorder) llrb ⇒'a ⇒ 'a llrb ⇒'a llrb ⇒ 'a llrb "

definition pre_invsplit'::" 'a llrb ⇒'a ⇒ 'a llrb ⇒ 'a llrb" [2 lines]

interpretation llrb_op pre_invl' pre_invr' pre_invsplit'
done
```

图 1-7　文件 proof_delete.thy 中的第 72-87 行

③ 实例化函数的正确性验证

ins'等价于 ins，del'等价于 del，如图 1-8 和 1-9 所示，我们证明了 ins'和 ins、del'和 del 之间的等价性。

proof_delete.thy (%USERPROFILE%\Desktop\LLRB_PROOF_12.4\)

```
lemma locins_eq_ins:"ins' x t = ins x t"
  apply(induct t)
  apply simp
  apply(case_tac "x2")
  apply(case_tac "b")
  apply (simp only: ins'.simps ins.simps)
  by (metis ins'.simps(2) ins.simps(2) baliL'_eq_baliL baliR'_eq_baliR)
```

图 1-8　文件 proof_delete.thy 中的第 178-184 行

proof_delete.thy (%USERPROFILE%\Desktop\LLRB_PROOF_12.4\)

```
lemma locdel_eq_del:"del' x t = del x t"
  apply(induct t)
  apply simp
  apply(case_tac "x2")
  apply (simp only: del'.simps del.simps)
  by (simp add: pre_invl'_def pre_invr'_def pre_invsplit'_def)
```

图 1-9　文件 proof_delete.thy 中的第 96-101 行

依据 LLRB 的特性，我们还需将 LLRB 的插入(ins)和删除(del)进行染黑操作（paint Black），从而得到函数 insert 和 delete，最后证明了 insert 和 delete 的正确性，至此我们完成了 LLRB 插入和删除操作的正确性证明，如图 1-10 所示。

proof_insert.thy (%USERPROFILE%\Desktop\LLRB_PROOF_12.4\)

```
definition insert :: "'a::linorder ⇒ 'a llrb ⇒ 'a llrb" where
"insert x t =  paint Black(ins x t)"
```

proof_delete.thy (%USERPROFILE%\Desktop\LLRB_PROOF_12.4\)

```
definition delete :: "'a::linorder ⇒ 'a llrb ⇒ 'a llrb" where
"delete x t = paint Black (del x t)"
```

proof_insert.thy (%USERPROFILE%\Desktop\LLRB_PROOF_12.4\)

```
theorem llrb_insert: "llrb t ⟹ llrb (insert x t)"
  by(metis invc_insert invh_insert llrb_def color_paint_Black insert_def)
```

proof_delete.thy (%USERPROFILE%\Desktop\LLRB_PROOF_12.4\)

```
theorem llrb_delete: "llrb t ⟹ llrb (delete x t)"
  by(metis color_paint_Black delete_def invc_delete invh_delete llrb_def)
```

图 1-10　insert 和 delete 的正确性验证

问题 4：论文图 4 中所提及的函数 split max 对应脚本的位置在哪？

对于删除操作，需要用待删除节点子树的最大元素或最小元素去替换待删除节点，因此本文给出了分离函数引理集 l_8 和 l_9(bst_split)，其中引理 l_8 所提及的 split_max 函数适用于 AVL 树、AA 树等，引理 l_9 所提及的 split_min 函数适用于非平衡树、RBs、2-3-4 树、1-2 兄弟树等。（论文第 8 页第三段也做了相应的解释）

本文关注的 LLRB 算法属于 RBs 类，只需使用 l_9，因此在对 LLRB 进行函数式建模和证明时使用了 split_min 函数，所以在我们的证明脚本中并未出现 split_max 函数。

问题 5：论文图 4 中所提及的 balance_fun 对应脚本的位置在哪？

balance_fun 表示为了维持二叉搜索树附加性质在进行插入、删除操作时所定义的泛化平衡函数，其参数均为 l,a,r，可实例化为具体函数（对应于论文的第 8 页第二段）。对 LLRB 算法进行证明时，balance_fun 可实例化为一系列平衡函数，比如 baliL、baliR、rightredB 等等。

我们以**引理集 l_7** 为例: inorder (**balance_fun** l a r) = inorder l @ a # inorder r

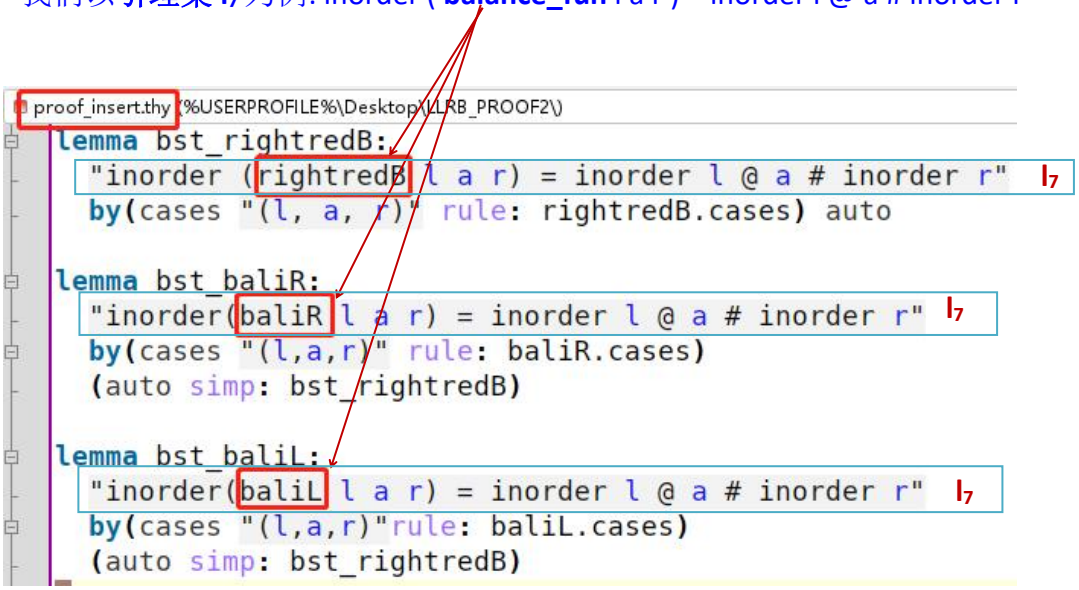

图 1-11　文件 proof_insert.thy 中的第 68-80 行

问题 6：论文第五章引理 13，14，15 对应脚本的位置在哪？

引理 13-15 如图 1-12 所示，位于文件 proof_insert.thy 中：

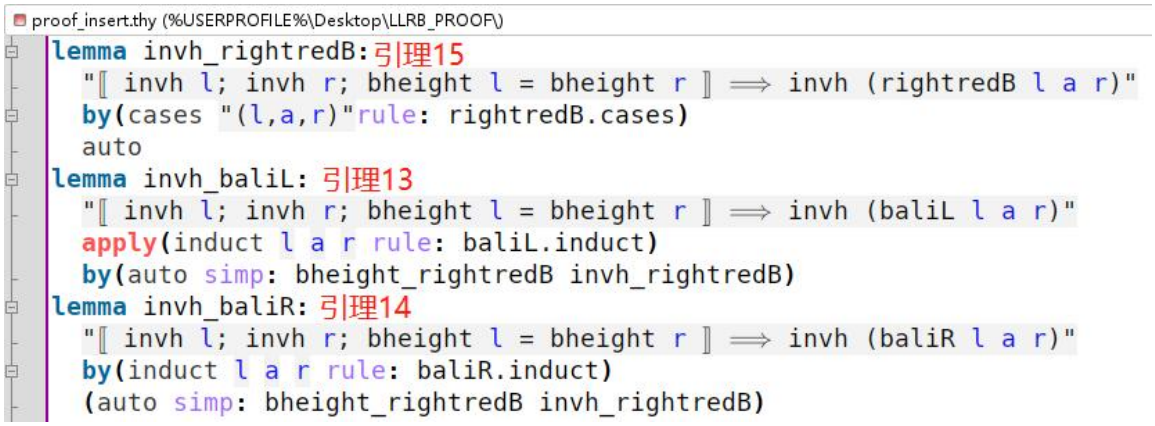

图 1-12　文件 proof_insert.thy 中的第 128-141 行

问题 6：论文第五章中定理 4 所对应的脚本位置在哪？

定理 4 对应的所有证明脚本在文件 proof_delete.thy 中，包含了 16 个辅助引理，如图 1-13 所示。

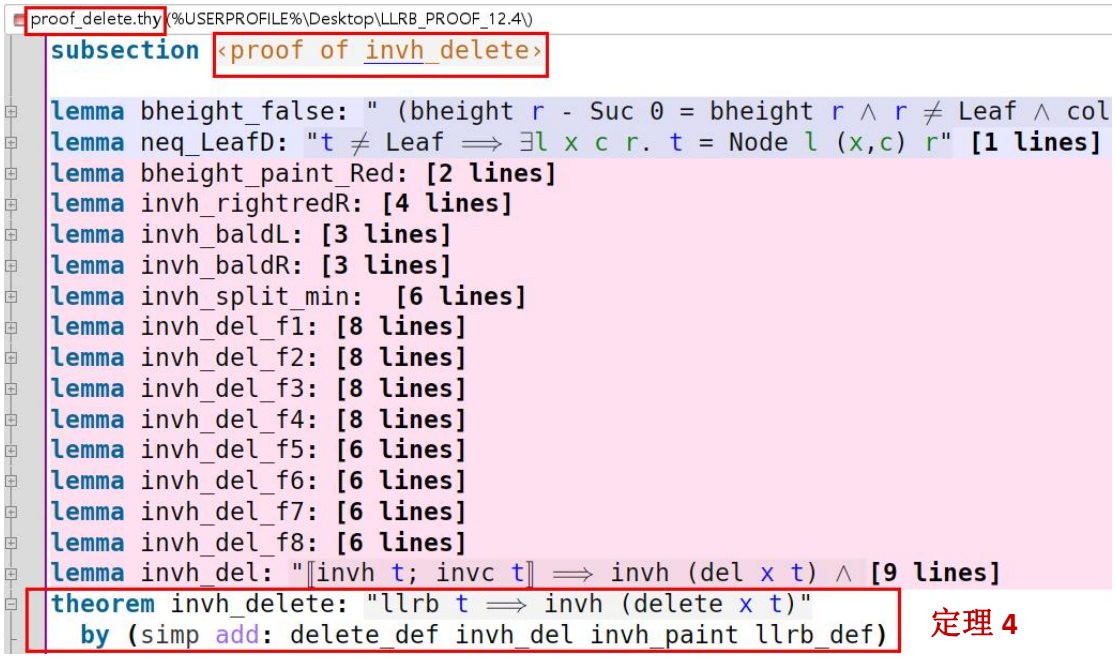

图 1-13　定理 4 的证明脚本

问题 7：论文中所有以 invh 为前缀的引理位置在哪？

文中所有以 invh_为前缀的引理位于 proof_insert.thy 和 proof_delete.thy 中，如图 1-14 和 1-15 所示。

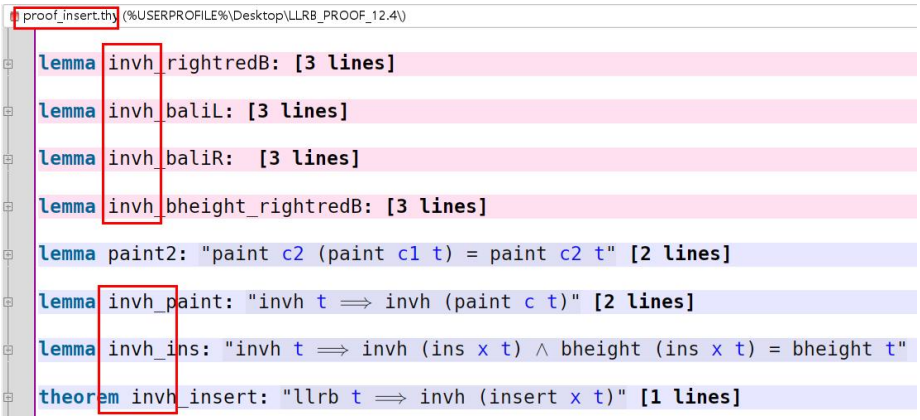

图 1-14　文件 proof_insert.thy 中的第 128-161 行

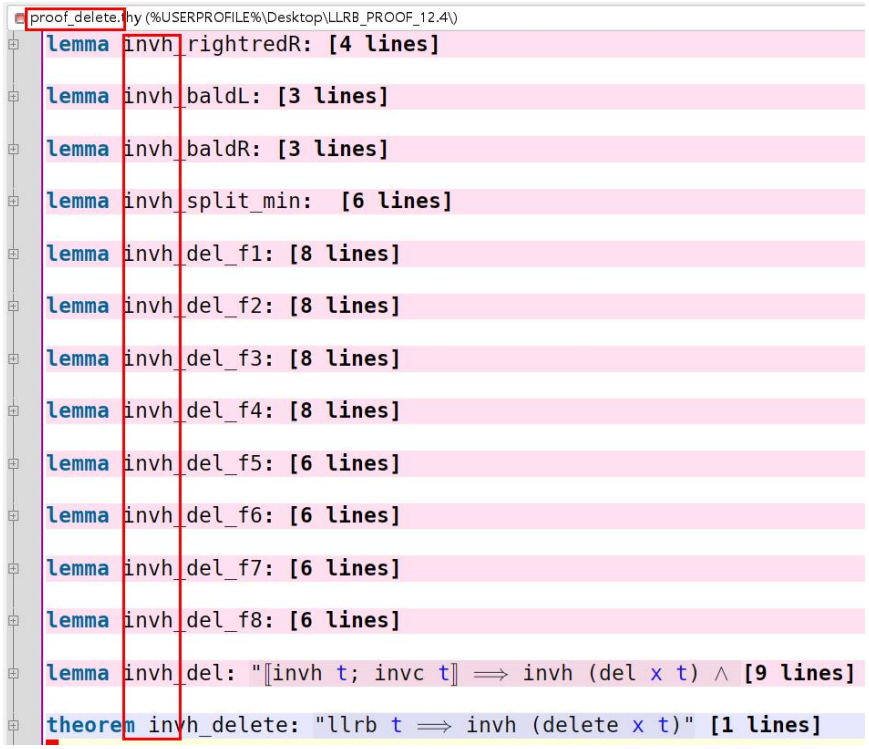

图 1-15　文件 proof_delete.thy 中的第 237-345 行

问题 8：论文中有无验证脚本地址？

验证脚本地址：https://github.com/Criank/LLRB_proof/tree/master(对应论文的第 5 章节的第一段)