上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

上海交通大学计算机科学与工程系
SJTU-CSE

VST-IDE：交互式程序验证工具（分离逻辑求解）

本科毕业设计答辩

唐亚周 519021910804 tangyazhou518@sjtu.edu.cn
指导老师：曹钦翔

2023 年 5 月

目录

第 1 节

绪论

第 1 节

绪论

第 1.1 小节

研究背景

研究背景

- 保障软件的正确性和安全性是软件开发过程中的重要环节。

研究背景

- 保障软件的正确性和安全性是软件开发过程中的重要环节。
- 形式化验证（formal verification）是指通过使用形式化方法来验证软件系统是否符合其规范或需求。

研究背景

- 保障软件的正确性和安全性是软件开发过程中的重要环节。
- 形式化验证（formal verification）是指通过使用形式化方法来验证软件系统是否符合其规范或需求。
- 定理证明（theorem proving）是形式化验证的一种方法，它将程序和系统的正确性表达为数学命题，然后使用逻辑推导的方式证明正确性。

研究背景

- 保障软件的正确性和安全性是软件开发过程中的重要环节。
- 形式化验证（formal verification）是指通过使用形式化方法来验证软件系统是否符合其规范或需求。
- 定理证明（theorem proving）是形式化验证的一种方法，它将程序和系统的正确性表达为数学命题，然后使用逻辑推导的方式证明正确性。

软件测试	程序验证
效率较高	效率较低
自动化程度较高	自动化程度较低
对测试人员要求较低	对验证人员要求较高
无法保证程序的完全正确性	可以保证程序的完全正确性

本项目解决的主要问题

- 主要目的：降低 C 语言程序验证的门槛和成本，并且尝试将验证结合到开发过程中，帮助软件开发者提高验证效率，实现“开发即安全”。
- 验证重点：C 语言的内存安全性（memory safety），比如对于内存泄漏、空指针引用、非法内存访问等问题的检测。

第 1 节

绪论

第 1.2 小节

相关技术介绍

霍尔逻辑

霍尔逻辑（Hoare Logic）[1]的中心特征是霍尔三元组（Hoare triple）：

$$\{P\}C\{Q\},$$

其中，

- P 和 Q 是断言，C 是程序指令。
- 如果断言 P 在程序 C 执行前为真，那么程序 C 执行后断言 Q 为真。
- 这里我们也把 P 称为前置条件（Precondition），Q 称为后置条件（Postcondition）。

[1] HOARE C A R. An axiomatic basis for computer programming[J]. Communications of the ACM, 1969, 12(10): 576-580.

分离逻辑

分离逻辑[2]是霍尔逻辑的扩展，它能够直观地分析程序中复杂的动态内存变化。
分离逻辑引入了 4 种新的断言形式，以细化对计算机系统存储状态的描述：

$$
\begin{aligned}
\langle\ \text{assert}\ \rangle ::= {} & \cdots \\
\mid\ & \mathbf{emp} && \text{空堆} \\
\mid\ & \langle \text{exp} \rangle \mapsto \langle \text{exp} \rangle && \text{单堆} \\
\mid\ & \langle\ \text{assert}\ \rangle * \langle\ \text{assert}\ \rangle && \text{分离合取} \\
\mid\ & \langle\ \text{assert}\ \rangle -\!\!* \langle\ \text{assert}\ \rangle && \text{分离蕴含}
\end{aligned}
$$

[2]REYNOLDS J C. Separation logic: A logic for shared mutable data structures[C]//Proceedings 17th Annual IEEE Symposium on Logic in Computer Science. 2002: 55-74.

第 2 节

整体框架

第 2 节

整体框架

第 2.1 小节

项目概述

项目概述

- 用户可以在编辑器中编写 C 语言程序，同时可以在注释中编写断言，从而在开发过程中实现对程序的验证。
- 主要优势：自动化程度更高。用户只需要编写少量程序断言，而不需要编写证明代码，就可以实现验证。
- 分为三个部分：编译前端、符号执行和基于分离逻辑的蕴含关系检验。

```
1 // 函数的前置条件
2 代码A
3 // 程序断言A
4 代码B
5 // 程序断言B
6 ...
7 // 函数的后置条件
```

❶ 前置条件 $\xrightarrow{\text{对代码 A 做符号执行}}$ 程序状态 A

```
1 // 函数的前置条件
2 代码A
3 // 程序断言A
4 代码B
5 // 程序断言B
6 ...
7 // 函数的后置条件
```

1. 前置条件 $\xrightarrow{\text{对代码 A 做符号执行}}$ 程序状态 A
2. 检验：程序状态 A $\vdash$ 程序断言 A 是否成立

```
1 // 函数的前置条件
2 代码A
3 // 程序断言A
4 代码B
5 // 程序断言B
6 ...
7 // 函数的后置条件
```

1. 前置条件 $\xrightarrow{\text{对代码 A 做符号执行}}$ 程序状态 A
2. 检验：程序状态 A $\vdash$ 程序断言 A 是否成立
3. 程序断言 A $\xrightarrow{\text{对代码 B 做符号执行}}$ 程序状态 B

```
1 // 函数的前置条件
2 代码A
3 // 程序断言A
4 代码B
5 // 程序断言B
6 ...
7 // 函数的后置条件
```

1. 前置条件 $\xrightarrow{\text{对代码 A 做符号执行}}$ 程序状态 A
2. 检验：程序状态 A $\vdash$ 程序断言 A 是否成立
3. 程序断言 A $\xrightarrow{\text{对代码 B 做符号执行}}$ 程序状态 B
4. 检验：程序状态 B $\vdash$ 程序断言 B 是否成立

```
1 // 函数的前置条件
2 代码A
3 // 程序断言A
4 代码B
5 // 程序断言B
6 ...
7 // 函数的后置条件
```

1. 前置条件 $\xrightarrow{\text{对代码 A 做符号执行}}$ 程序状态 A
2. 检验：程序状态 A $\vdash$ 程序断言 A 是否成立
3. 程序断言 A $\xrightarrow{\text{对代码 B 做符号执行}}$ 程序状态 B
4. 检验：程序状态 B $\vdash$ 程序断言 B 是否成立
5. ...
6. 检验：最后的程序状态 $\vdash$ 后置条件 是否成立

```
1 // 函数的前置条件
2 代码A
3 // 程序断言A
4 代码B
5 // 程序断言B
6 ...
7 // 函数的后置条件
```

1. 前置条件 $\xrightarrow{\text{对代码 A 做符号执行}}$ 程序状态 A
2. 检验：程序状态 A $\vdash$ 程序断言 A 是否成立
3. 程序断言 A $\xrightarrow{\text{对代码 B 做符号执行}}$ 程序状态 B
4. 检验：程序状态 B $\vdash$ 程序断言 B 是否成立
5. ...
6. 检验：最后的程序状态 $\vdash$ 后置条件 是否成立

将检验过程以 VST 证明的格式输出，
从而可以通过 Coq 验证我们对于 程序状态 $\vdash$ 程序断言 的证明是否正确。

第 2 节

整体框架

第 2.2 小节

蕴含关系检验

基于分离逻辑的蕴含关系检验

在 VST 中，断言有着如下的格式[3]：

$$\text{EX}\ldots\text{PROP}(\ldots)\text{LOCAL}(\ldots)\text{SEP}(\ldots).$$

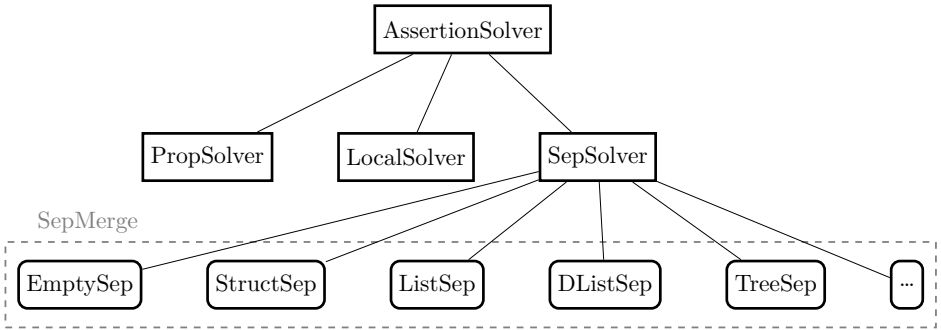
其中，

- EX：即 Exists，用于表示该断言中的存在变量；
- PROP：即 Proposition，用于表示独立于程序状态的真命题；
- LOCAL：即 Local，用于表示 C 语言中的局部变量的值，即程序变量与逻辑变量的映射；
- SEP：即 Separation，用于表示分离逻辑中的空间断言，即内存空间中的状态。

[3] ANDREW W. APPEL L B, et al. Verifiable C: vol. 5[M]. Ed. by PIERCE B C. Electronic textbook, 2023.

单个断言蕴含单个断言

对于 Sep 求解器进行了模块化的设计



多个断言蕴含多个断言

符号执行过程中如果遇到控制语句就会根据条件进行分支，因此在实际的验证过程中，我们的程序状态可能为多个断言的析取。

在这种情况下，蕴含关系求解器需要证明以下的蕴含关系：

$$Q'_1 \vee Q'_2 \vee \cdots \vee Q'_n \vdash Q_1 \vee Q_2 \vee \cdots \vee Q_m,$$

这里我们要求对于任意的 Q'_i，都存在一个 Q_j 使得 $Q'_i \vdash Q_j$。
多个断言蕴含多个断言的情况转化为单个断言蕴含单个断言的情况。

第 3 节

实现细节

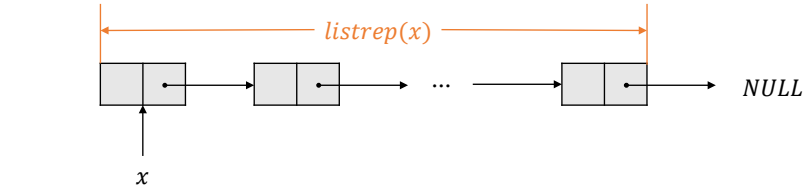
第 3 节

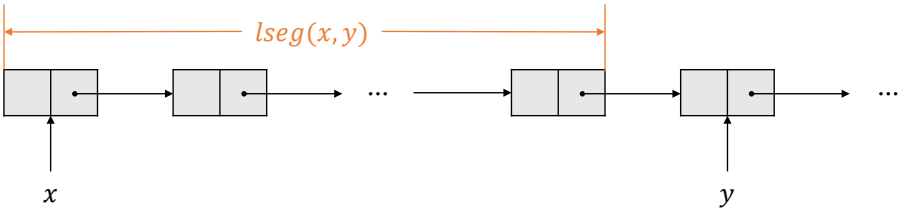
实现细节

第 3.1 小节

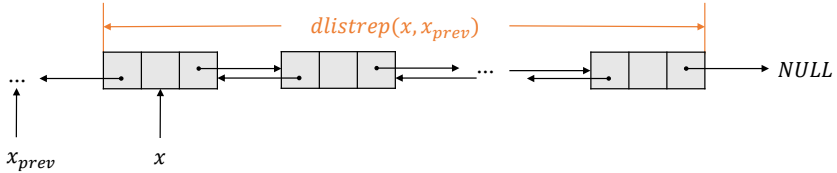
定义

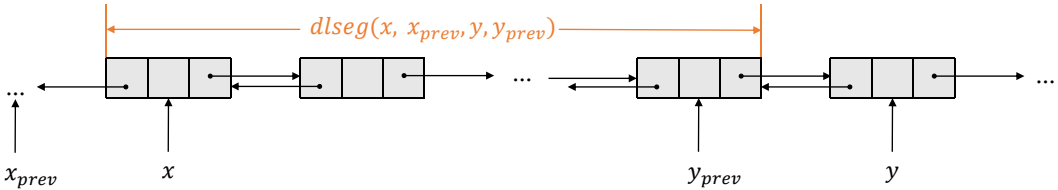
单链表谓词定义



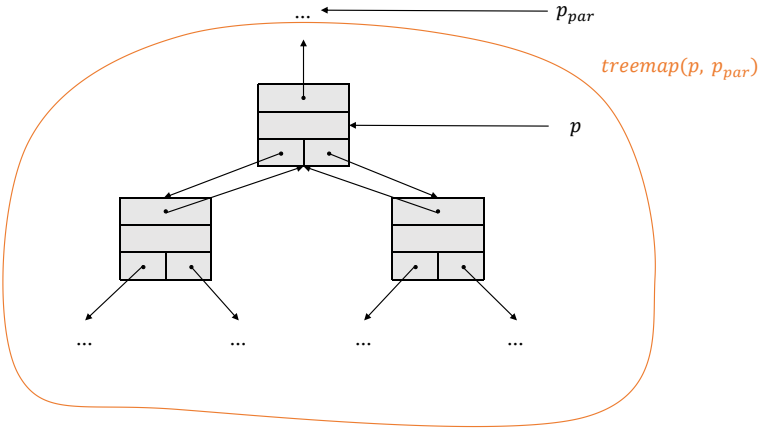


双链表谓词定义

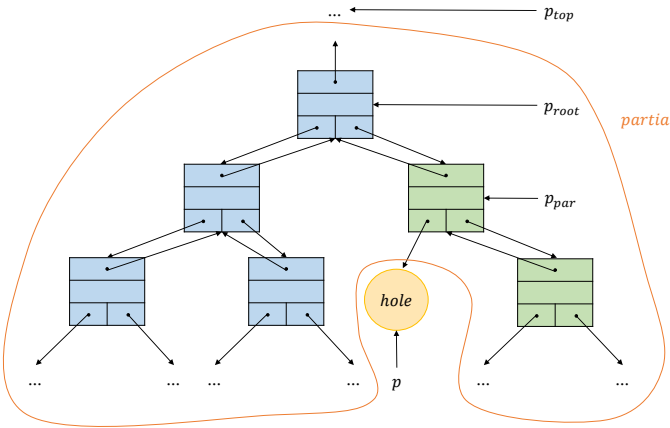

dlistrep(x, x_prev)
NULL
x_prev
x


dlseg(x, x_prev, y, y_prev)
x_prev
x
y_prev
y

二叉树谓词定义



二叉树谓词定义



$partial\_treemap(p, p_{par}, p_{root}, p_{top})$

第 3 节
实现细节

第 3.2 小节
求解流程

预处理

对蕴含关系左右两边的断言做预处理：

1. 空堆（emp）的筛除；
2. 纯事实（pure fact）的获取；
3. 存在变量的实例化。

Prop/Local/Sep 的求解器

- Prop 求解器：主要应用于求解结束时对右边断言的 Prop 证明，以及求解过程中对于一些临时产生的 Prop 的证明。
- Local 求解器：主要应用与 Local 相关存在变量的实例化，以及 Local 的“消去”。
- Sep 求解器：主要应用与 Sep 相关存在变量的实例化，以及 Sep 的“消去”。

消去：如果在蕴含关系的两边发现了一模一样的部分，那么就可以把问题转化为求解“两边都去除该部分之后”的蕴含关系。

$$\textbf{Frame}\quad \frac{\{P\}\mathtt{C}\{Q\}}{\{P * F\}\mathtt{C}\{Q * F\}}$$

拆分

- 一般来说，蕴含关系左右两边的 Sep 并不会完全相同，需要我们对右边的 Sep 进行变换之后再将它们输入到 Sep 求解器，才能够尽可能地“消去”。
- 这里的变换主要是对用户自定义谓词的拆分，拆分得到的结果是与用户所定义的结构体中的数据成员（field）相关的。

算法：对于右边 Sep 的每一个自定义谓词 P，去左边 Sep 中寻找同一地址的单堆空间断言，并判断该断言是否符合 P 的某种字段的定义。如果符合则按照定义拆分 P。

举例：$(x.head \mapsto a) \vdash listrep(x)$，识别到左边的 $x.head \mapsto a$ 是右边 $listrep(x)$ 的 $head$ 字段，那么右边会被拆分为 $EX\ y, (x.head \mapsto a) * (x.tail \mapsto y) * listrep(y)$。

化简

一些空间断言虽然满足谓词定义中空堆的条件，但也有可能并不是空堆。

举例：单链表段的谓词定义如下：

$$lseg(x, y) := (x = y \&\& emp) || lseg(x \to tail, y)$$

但 $lseg(x, x)$ 并不一定是 emp，也可能为 $EX\ y, lseg(x, y) * (y \mapsto x)$。
判定的方法：

- 因为分离合取是将内存分为“互不相交”的几个部分，所以一个地址所表示的内存空间不能被多次描述。
- 如果整个空间断言为 $lseg(x, x) * (x \mapsto a)$，那么对于地址 x 已经有一个空间断言对其进行描述。如果 $lseg(x, x)$ 不为空堆的话，就相当于对 x 再次进行了描述，违反了分离逻辑的规则。

第 3 节

实现细节

第 3.3 小节

证明规则生成

证明规则生成

为了验证求解器的求解是否正确，我们在求解过程中保存了所使用到的证明规则，并且在求解结果中输出。

对于 Prop/Local/Sep，我们定义了不同的证明规则。在检验结束后，我们会对证明规则进行筛选，然后将其转换为 Coq 中的证明代码。

我们在 Coq 中根据证明规则定义了一些引理，因此输出时只需要使用证明过程对应的引理即可。

第 4 节

总结

测试样例

数据结构类型	操作	验证状态
单链表	反转	成功
单链表	遍历	成功
单链表	连接两个链表	成功
双链表	反转	成功
双链表	连接两个链表	成功
双链表	直接删除结点	成功
双链表	遍历删除结点	成功
二叉树	插入/删除/查找结点	调试中
其它	交换两个变量的值	成功

总结和展望

本项目提出了一种交互式程序验证工具 VST-IDE，它允许用户以注释的形式进行谓词的自定义和断言的编写，采用符号执行和蕴含关系检验的方式，降低程序验证的门槛。

在未来，VST-IDE 的蕴含关系检验部分需要进一步完善和改进算法，以支持更复杂的数据结构和算法的验证；同时，我们也在关注功能正确性（functional correctness）的验证，比内存安全性验证的难度更大。

参考文献

[1] HOARE C A R. An axiomatic basis for computer programming[J]. Communications of the ACM, 1969, 12(10): 576-580.

[2] REYNOLDS J C. Separation logic: A logic for shared mutable data structures[C]//Proceedings 17th Annual IEEE Symposium on Logic in Computer Science. 2002: 55-74.

[3] ANDREW W. APPEL L B, CAO Q. Verifiable C: vol. 5[M]. Ed. by PIERCE B C. Electronic textbook, 2023.

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

谢 谢

唐亚周 519021910804 tangyazhou518@sjtu.edu.cn
指导老师：曹钦翔
VST-IDE：交互式程序验证工具（分离逻辑求解）