软件科学基础

Separation Logic: a Quick Look

熊英飞
北京大学

扩展霍尔逻辑验证指针

```
void swap(int *x, int* y) {
    *x = (*x)^(*y);
    *y = (*x)^(*y);
    *x = (*x)^(*y);
}
```

下面的霍尔三元组成立吗?

$$\{*\,\mathrm{x} = a \land *\,\mathrm{y} = b\}\mathrm{swap}(\mathrm{x}, \mathrm{y})\{*\,\mathrm{x} = b \land *\,\mathrm{y} = a\}$$

扩展霍尔逻辑验证指针

- 不成立，因为x和y可能是别名（指向同一块地址）
- 一个成立的霍尔三元组如下所示

$$\{x \neq y \rightarrow * \; \mathrm{x} = a \wedge * \; \mathrm{y} = b\}\mathrm{swap}(\mathrm{x}, \mathrm{y})\{x \neq y \rightarrow * \; \mathrm{x} = b \wedge * \; \mathrm{y} = a\}$$

- 对于指针程序的验证需要反复论证别名
  - 如链表需要无环，两个链表连接不能连接自己的一部分
- 能否把别名的论证变成逻辑的一部分?

分离逻辑Separation Logic

- 扩展了霍尔逻辑，将“无别名”作为逻辑符号的一部分
- 源于John C. Reynolds, Peter O‘Hearn, Samin Ishtiaq and Hongseok Yang等人在99-01年期间发表的四篇论文
- 是Facebook的程序分析工具Infer的理论基础（理论上）
- 本世纪程序验证上最重要的工作之一
    - 获得2016年的CAV Award和哥德尔奖

分离逻辑概念

- Store：从变量名到变量值的映射
  - 表示栈
- Heap：从内存地址到内存值的部分映射
  - 表示堆
  - $h_1 \perp h_2$表示两个堆不交，即定义域交集为空
  - $h_1 \cup h_2$表示合并两个不交的堆

分离逻辑断言

- 语法

$$
\begin{array}{lcc}
Ass & := & \mathbf{emp} \\
 & | & Exp_1 \mapsto Exp_2 \\
 & | & Ass_1 * Ass_2 \\
 & | & Ass_1 -\!\!* \; Ass_2 \\
 & | & \text{其他一阶逻辑式子} \\
Exp & := & \text{程序表达式}
\end{array}
$$

- 语义
  - 三元关系$s, h \vDash P$表示谓词P在Store s和Heap h上成立

emp

- $s, h \vDash \mathbf{emp}$
  - 表示h为空，对所有内存地址都未定义

$Exp_1 \mapsto Exp_2$

- $s, h \vDash e \mapsto e'$
  - 表示地址e保存了值e'，即$h([e]_s) = [e']_s$且h对$[e]_s$以外的值都没有定义，其中$[e]_s$表示e在s上求值的结果
  - 如$x \mapsto$1表示指针x所指向的地址保存了值1，且堆中没有其他地址
  - 如$x \mapsto n + 1$表示指针x所指向的地址保存的值等于变量n保存的值加上1，且堆中没有其他地址

$Ass_1 * Ass_2$ （分离合取、星号）

- $s, h \vDash P * Q$
  - $\exists h_1, h_2$，使得$h = h_1 \cup h_2 \land h_1 \perp h_2$
    - $s, h_1 \vDash P$
    - $s, h_2 \vDash Q$
  - 即P和Q都成立，但其中地址不相交
  - 分离逻辑中最重要的逻辑符号
  - 如$x \mapsto a * y \mapsto b$说明x和y不是别名
  - 如：下面递归定义了判断p是否为链表的谓词，*用来保证链表不会出现环
    - $list(p) = p = null \to \mathbf{emp}$
      $\land\, p \neq null$
      $\to \exists a, p', p \mapsto \{value{:}\, a, next{:}\, p'\} * list(p')$
    - 练习：用分离合取定义二叉树的谓词
  - 性质：$P = \mathbf{emp} * P = P * \mathbf{emp}$

$Ass_1 -\!* \ Ass_2$（分离蕴含、魔法棒）

- $s, h \vDash P -\!* \ Q$
  - $\forall h', h' \perp h \wedge s, h' \vDash P \ \longrightarrow s, h' \cup h \vDash Q$
  - 即给h补上一块满足P的堆就能满足Q
  - 如$(x \mapsto 1) -\!* \ Q$表示当前堆不包括x，但加上一个x指向1的堆就满足Q

- 分离合取和分离蕴含具有一些普通合取和蕴含的性质
  - $\frac{\vDash P \wedge (P \rightarrow Q)}{\vDash Q}$ $\qquad$ $\frac{s,h \vDash P * (P -\!* Q)}{s,h \vDash Q}$

扩展霍尔三元组

- {P}c{Q}
  - 表示对任意满足P的s和h，如果执行语句c之后得到了s’和h’，那么s’和h’满足Q

- 如：
  - $\{x \mapsto a * y \mapsto b\}$swap(x,y)$\{x \mapsto b * y \mapsto a\}$

推导规则-新增语句相关

- Store: $\{x \mapsto -\}$*x=v$\{x \mapsto v\}$
- Load: $\{x \mapsto v\}$a=*x$\{a = v \wedge x \mapsto v\}$
- Alloc: $\{\mathbf{emp}\}$x=malloc()$\{x \mapsto -\}$
- AllocFail: $\{\mathbf{emp}\}$x=malloc()$\{x \mapsto - \vee x = 0\}$
- DeAlloc: $\{x \mapsto -\}$free(x)$\{\mathbf{emp}\}$

- "$x \mapsto -$"表示"$\exists v, x \mapsto v$"

推导规则-Frame规则

- Frame:

- $$\frac{\{P\}c\{Q\}}{\{P*R\}c\{Q*R\}}$$

完整性

- 以上推导规则，加上断言上的公理（本课程跳过）、部分霍尔逻辑规则和一阶逻辑规则，可以推导出所有的为真的三元组

证明举例1

- 证明定理：$\{x \mapsto a * y \mapsto b\}$t=*x;*x=*y;*y=t;$\{x \mapsto b * y \mapsto a\}$
- 根据Load可得
  - $\{x \mapsto a\} t =* x \{t = a \wedge x \mapsto a\}$
- 注意$t = a$不涉及到堆，根据Consequence可得
  - $\{x \mapsto a\} t =* x \{(t = a \wedge \mathbf{emp}) * x \mapsto a\}$
- 再根据Frame可得
  - $\{x \mapsto a * y \mapsto b\} t =* x \{(t = a \wedge \mathbf{emp}) * x \mapsto a * y \mapsto b\}$
- 根据Load和Store可得
  - $\{x \mapsto - * y \mapsto b\}$*x=*y$\{x \mapsto b * y \mapsto b\}$
- 根据Frame和Consequence可得
  - $\{(t = a \wedge \mathbf{emp}) * x \mapsto a * y \mapsto b\}$*x=*y$\{(t = a \wedge \mathbf{emp}) * x \mapsto b * y \mapsto b\}$
- 类似可得
  - $\{(t = a \wedge \mathbf{emp}) * x \mapsto b * y \mapsto b\}$*y=t$\{x \mapsto b * y \mapsto a\}$
- 根据Seq，原命题得证

证明举例2

- void delete(p) {
```
void delete(p) {
  if (p = null) return;
  else {
    delete(p->next);
    free(p);
  }
}
```
- 证明：$\{list(p)\}$delete(p)$\{\mathbf{emp}\}$

证明举例2

- 先考虑then分支，根据Skip可得
  - $\{\mathbf{emp}\}skip\{\mathbf{emp}\}$
- 再根据Consequence可得
  - $\{list(p) \wedge p = null\}skip\{\mathbf{emp}\}$
- 再考虑else分支，根据DeAlloc可得
  - $\{p \mapsto -\}\mathrm{free(p)}\{\mathbf{emp}\}$
- 根据归纳假设可得
  - $\{list(p \rightarrow next)\}$delete(p->next)$\{\mathbf{emp}\}$
- 从上面两项，根据Frame可得
  - $\{list(p \rightarrow next) * p \mapsto -\}$delete(p->next)$\{p \mapsto - * \mathbf{emp}\}$
  - $\{p \mapsto - * \mathbf{emp}\}\mathrm{free(p)}\{\mathbf{emp} * \mathbf{emp}\}$
- 再根据Consequence可得
  - $\{(list(p) \wedge p \neq null)\}$delete(p->next)$\{p \mapsto - * \mathbf{emp}\}$
  - $\{p \mapsto - * \mathbf{emp}\}\mathrm{free(p)}\{\mathbf{emp}\}$
- 最后根据If，原命题得证

教材和分离逻辑相关内容

- 教材第5卷：一套用分离逻辑验证C语言的证明系统VST
- 教材第6卷：一套基于Ocaml程序的验证介绍分离逻辑的教程

VST

- 用分离逻辑验证C语言的证明系统
- VST的逻辑是higher-order impredicative concurrent separation logic
  - separation logic——用于处理指针
  - concurrent separation logic——用于处理并行
  - higher-order impredicative program logic——用于处理函数指针、面向对象重载等
- 上海交通大学曹钦翔老师为主力开发和维护人员之一
- 可以高效验证实际的C程序

曹钦翔
上海交大副教授
CMO、NOI双全国第一
北大哲学本科、普林斯顿博士

VST

- Decorated Program面向实际C程序的终极强化版
  - 提供Coq数据结构表示C程序的AST
  - 提供Coq类型表示断言和霍尔三元组
  - 提供Coq证明策略用于证明霍尔三元组
- 重新在Coq中定义出一套新的面向C的证明语言

VST验证举例——程序

```c
#include <stddef.h>

unsigned sumarray(unsigned a[], int n) {
  int i; unsigned s;
  i=0;
  s=0;
  while (i<n) {
    s+=a[i];
    i++;
  }
  return s;
}
```

```c
unsigned four[4] = {1,2,3,4};

int main(void) {
  unsigned int s;
  s = sumarray(four,4);
  return (int)s;
}
```

VST验证举例——规约

数组地址　内存　数组内容　数组大小　辅助函数

```
Definition sum_Z : list Z → Z := fold_right Z.add 0.

Definition sumarray_spec : ident × funspec :=
DECLARE _sumarray
  WITH a: val, sh : share, contents : list Z, size: Z
  PRE [ tptr tuint, tint ]
    PROP (readable_share sh; 0 ≤ size ≤ Int.max_signed;
              Forall (fun x ⇒ 0 ≤ x ≤ Int.max_unsigned) contents)
    PARAMS (a; Vint (Int.repr size))
    SEP (data_at sh (tarray tuint size) (map Vint (map Int.repr contents)) a)
  POST [ tuint ]
    PROP () RETURN (Vint (Int.repr (sum_Z contents)))
    SEP (data_at sh (tarray tuint size) (map Vint (map Int.repr contents)) a).
```

参数

数组大小为正

数组元素为正

关联参数1的三个值

声明规约中要用的变量

返回值要求

VST验证举例——证明片段

```
Lemma body_sumarray: semax_body Vprog Gprog f_sumarray sumarray_spec.
Proof.
    forward.
    forward.
    forward_while
      (EX i: Z,
         PROP (0 ≤ i ≤ size)
         LOCAL (temp _a a;
                       temp _i (Vint (Int.repr i));
                       temp _n (Vint (Int.repr size));
                       temp _s (Vint (Int.repr (sum_Z (sublist 0 i contents)))))
         SEP (data_at sh (tarray tuint size) (map Vint (map Int.repr contents)) a)).

    - hint.
```

定理

根据霍尔逻辑规则计算第一条语句的最强后条件

设置循环不变式

让系统提示下一步证明指令

循环不变式产生多个子目标