Лекция 5. Спецификация и верификация программ с указателями

Цель лекции

Узнать особенности спецификации и верификации функций, оперирующих с указателями и массивами.

Содержание

1 Спецификация сортировки

2 Спецификация аллоцирования и деаллоцирования памяти

Основные конструкции

- Валидность диапазона указателей лучше специфицировать без квантора всеобщности: `\valid(array + (0 .. size - 1));`
- Постусловие имеет дело с двумя состояниями памяти: до вызова функции и после вызова функции: *метки памяти* - `Pre` и `Post`
- Чтобы разыменовать указатель, надо указать состояние памяти: `\at(expression, label)`
- У предиката метку памяти можно указать явно: `p{L}(n)`

Пример: сортировка выбором

- `sort_1.c` - спецификация сортировки
- `sort_2.c` - реализация сортировки выбором
- `sort_3.c` - доказательство safety
- `sort_4.c` - доказательство упорядоченности
- `sort_5.c` - доказательство перестановочности

Выводы из примера

- Солверам надо подсказывать, как нужно инстанцировать аксиомы
- Полезна бывает аксиома о том, что значение лоджика или предиката не изменится, если такая-то часть памяти между двумя метками не менялась
- Предикаты, аксиомы, леммы, лоджики могут иметь несколько меток памяти
- Можно задавать имя дополнительной метке памяти при помощи ghost
- В начале итерации цикла содержимое памяти надо вручную связывать с содержимым памяти до цикла, если в цикле есть присваивание в эту память

Содержание

1 Спецификация сортировки

2 Спецификация аллоцирования и деаллоцирования памяти

Модель памяти

- Все указатели делятся на классы эквивалентности — *блоки указателей*. Блок указателей — это все указатели, которые можно сравнивать между собой. Сравнение указателей разных блоков не определено (стандарт языка Си не разрешает сравнивать указатели, полученные разными выделениями памяти).
- В каждом блоке указателей есть один выделенный элемент — *базовый указатель*.
- Модель памяти сопоставляет каждому блоку указателей число — *размер*. Размер может быть:
  - ненулевым - можно разыменовать указатели от базового до базового + размер - 1
  - равным нулю - нельзя разыменовать ни один указатель, но можно вызвать функцию `free()` от базового указателя
  - отрицательным - разыменовать никакой нельзя, `free()` нельзя

Блоки указателей

- У указателей нет числового значения. Есть лишь смещение от базового указателя.
- Нельзя прибавлять по 1 к указателю одного блока и получить указатель из другого блока.
- Выделение и освобождение памяти - это смена размера блока. Блоки не появляются и не исчезают.
- Базовые адреса в блоке существуют независимо от того, какие размеры им сопоставлены в модели памяти.
- Классы эквивалентности указателей не меняются во время всей программы. Они вычисляются при помощи статического анализа (значения тех переменных, которые сравниваются в коде или спецификации).

Внутренние структуры

Модель памяти состоит из нескольких таблиц:

- таблицы блоков (`alloc_table`)
- таблицы тегов (`tag_table`)
- таблицы значений (`map from pointer to value type`)

Таблица блоков нужна для организации соответствия блоков указателей размерам. Таблица тегов — для организации динамических типов указателей. Таблица значений – для определения, на какое значение указывает указатель. Валидность проверяется по таблице блоков и тегов, разыменование делается при помощи таблицы значений.

Нормализация кода

Все типы приводятся к указателям на структуру с единственным полем (отсюда такие длинные имена типов в теории Why3, которую генерирует AstraVer).
Делается статический анализ для определения блоков указателей.
Для упрощения верификации делаются предположения:

- невыровненных указателей нет
- указатели разных типов не указывают в одну область памяти (или внутрь нее)
- указатели не преобразуются между типами указателей
- память неограничена

Предопределенные предикаты и лоджики

- `\freeable{L}(p)` - указатель p равен базовому указателю своего блока и в метке памяти L размер этого блока неотрицательный
- `\allocable{L}(p)` - указатель p равен базовому указателю своего блока и в метке памяти L размер отрицательный
- `\offset_min{L}(p)` - смещение от p до базового указателя блока указателей, которому принадлежит p
- `\offset_max{L}(p)` - значение `\offset_min{L}(p)` плюс размер блока указателей, которому принадлежит p, в метке памяти L минус 1, если этот размер неотрицательный

Пример

- `graph_1.c` — спецификация функций создания и удаления графа
- `graph_2.c` — определения функций
- `graph_3.c` — доказательство полной корректности (обратите внимание, как свойства массива `g->vertices` после первого цикла в функции создания сохранились по окончании второго цикла)