

Формальная спецификация и верификация программ.

Лекция 7.

Спецификация побочного эффекта программ  
(heap manipulations specification)



---

А.К.Петренко, А.В.Хорошилов,  
Е.В.Корныхин

МГУ ВМК, ИСП РАН

<http://sp.cmc.msu.ru/courses/fmsp>

Осень, 2012



# Указатели в языке Си

---

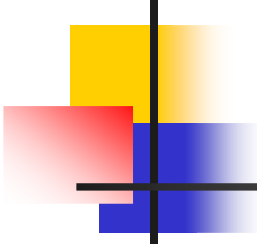
- `char *strcpy(char *dest, const char *src)`  
{  
    `char *d = dest;`  
    `while ( *d++ = *src++ );`  
    `return dest;`  
}



# Ссылки в языке Java

---

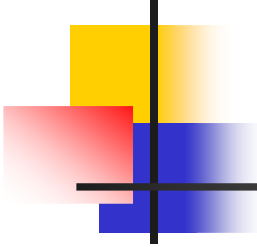
- `void execute(File f) throws Exception`  
{  
    `Parser parser = new Parser(f);`  
    `Tree tree = parser.parse();`  
    `if (tree != null) {`  
        `tree.interpret();`  
    }  
}



# Дефекты из-за некорректной работы с динамич-й памятью

---

- (перечислить)



# Дефекты из-за некорректной работы с динамич-й памятью

---

- null dereference
- read from/write to unallocated memory
- double free
- memory leak
- thread-unsafety
- **functional incorrectness**



# Функциональные дефекты

---

- нарушение инвариантов класса (неверная динамическая структура данных)
- запись в неразрешенную память
  - write x, read y; ... \*p = 10; → y = 10 !
- вызов функций в неразрешенном окружении



# Модель памяти

---

- часть семантики ЯП (как программа видит память, что может делать)
- какова модель памяти Java? какова модель памяти языка Си?

Xavier Leroy and Sandrine Blazy, Formal verification of a C-like memory model and its uses for verifying program transformations, 2007

Sascha Böhme and Michał Moskal, Heaps and Data Structures: A Challenge for Automated Provers, 2011



# Возвращаясь к Dafny

---

- модель памяти – отдельные объекты
- `modifies` clause
- `reads` clause
- `old`

K. Rustan M. Leino: Dafny: An Automatic Program Verifier for Functional Correctness, 2010





# Demo

---

- Однонаправленный список
  - <http://rise4fun.com/Dafny/vp2N>
- Какие встретились проблемы ?



# Возникшие трудности

---

- может ли contains зацикливаться ?
  - ацикличность списка
  
- как сформулировать постусловия методов ?
  - все значения, хранимые в списке



# Ещё пример

---

```
■ class List
{
...
public List() { ... }

public int len() {...}

}
```

```
■ static void test()
{
List A =
new List();
List B =
new List();

assert A.len()
== 0;
}
```

**fail !**





# Спецификация побочного эффекта подпрограмм (hear)

---

- топология структуры данных в hear
  - пример: ацикличность
- немодифицируемая область hear
  - пример: вся память вне того, что входит в данный список
- ограничения на результаты работы подпрограммы
  - пример: в список добавляется элемент



# Решение: подходы

---

- **explicit footprints**
- implicit footprints
- predefined topologies

John Hatcliff, Gary T. Leavens, K. Ristan M. Leino, Peter Müller, Matthew J. Parkinson: Behavioral interface specification languages. ACM Comput. Surv. 44(3): 16 (2012)

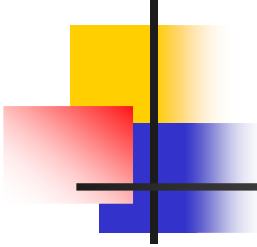


# Explicit footprint (dynamic frame)

---

- множество объектов, разрешенных для модификации (остальные модифицировать нельзя)
- фреймы не пересекаются → вызовы методов на них независимы

Ioannis T. Kassios. Dynamic Frames and Automated Verification.  
Tutorial presented at the 2nd COST Action IC0701 Training  
School in Limerick, Ireland, 2011



# Скелет класса с динамическим фреймом (1)

---

- ```
class List<T> { ...  
    ghost var footprint : set<List<T>>;  
    ...  
    method add(x : T)  
        modifies footprint;  
  
    function contains(x: T) : bool  
        reads footprint;  
}
```



# Swinging Pivots

---

- method test(A : List, B : List)  
requires A.footprint !! B.footprint;  
requires B.len() == 1; {  
    A.add(10);  
    assert B.len() == 1; **ok**  
    A.add(10);  
    assert B.len() == 1; **fail**  
}





# Swinging Pivots Restriction

---

- The value of any dynamic frame may be increased only by locations that are initially in some other dynamic frame or by newly allocated locations.
- ```
class List<T> { ...  
    method add(x : T)  
        modifies footprint;  
        ensures fresh(footprint – old(footprint));  
}
```

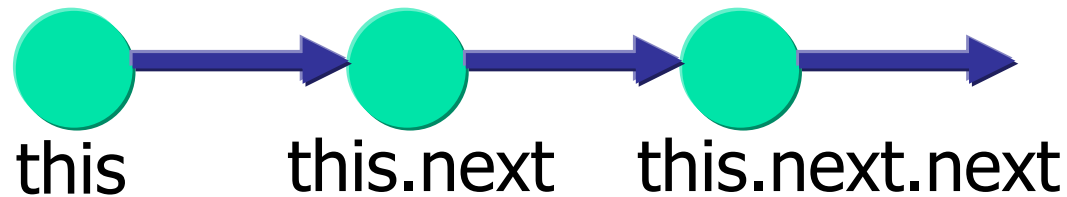
# Завершаемость рекурсивных функций с динамич. фреймом

- function contains(x : T) : bool  
reads this, footprint;  
{  
  head == x ||  
  (tail != null && tail.contains(x))  
}

**Error: failure to decrease termination measure**

**Error: insufficient reads clause to invoke function**

# Цепочка динамических фреймов - ацикличность



- $\text{this.footprint} = \{ \text{this.next}, \text{this.next.next}, \dots \}$
- $\text{this.next.footprint} = \{ \text{this.next.next}, \dots \}$
- forall  $x :: x \text{ in footprint} \ \&\& \ x.\text{next} \neq \text{null} \implies x.\text{next.footprint} < x.\text{footprint}$

**ЭТО ИНВАРИАНТНОЕ СВОЙСТВО !**



# Итоговая реализация List

---

- Добавить инвариант Valid
- Добавить поле «все элементы» (values) для спецификации contains
- <http://rise4fun.com/Dafny/bbYJu>



# Dynamic frame idiom in Dafny

```
■ class Coo {  
    ghost var footprint : set <object>;  
    function Valid () : bool  
        reads this , footprint ;  
    { this in footprint && ... }  
    constructor Init ()  
        modifies this ;  
        ensures Valid() && fresh( footprint - {this} ) ;  
    {  
        footprint := { this }; ...  
    }  
    method Mutate ( )  
        requires Valid();  
        modifies footprint;  
        ensures Valid() && fresh(footprint - old(footprint));  
    { ... }  
}
```