Total Pasta:

Unfailing Pointer Programs

Neil Mitchell, ndm AT cs.york.ac.uk

Department of Computer Science, University of York

Pasta – Linked List Example

```
list {
        nil();
        cons(int head, ptr tail);
}

-- inserts an element into an ordered list
insert(int i, ptr s) {
        while (s::cons && s->head < i) s = s->tail;
        if (s::nil || s->head > i) *s = *cons(i,copy(s));
}

main() {
        ptr r = nil();
        insert(1,r); insert(9,r);
        insert(2,r); insert(8,r);
}
```

Total Pasta Functions?

- Must not crash
  - `if (s::nil) s = s->tail;`
- Must terminate
  - `while (s::cons) s = s;`
- Don't need to worry about
  - arithmetic overflow (no addition in Pasta!)
  - recursion (also not in Pasta)
- Assume unbounded memory

Subtype checking

- Subtype annotations
  - `if (x::cons) …`
- Subtype assertions
  - `x->tail` requires `x::cons`
- Can use powerset to represent subtypes
  - Subtype(x) $\in$ {{cons,nil}, {nil}, {cons}, $\varnothing$}

Type assertions can be discharged by static checking

Termination Checking

- Only has a `while` statement to loop
- There must be one variable that is advanced down an acyclic path during every iteration
  - `while (s::cons) s = s->tail;`
- Requires an acyclic annotation
  - `list acyclic(tail) { … }`

My Approach

- B/Z inspired approach
  - Define postconditions for safety
  - Propagate backwards
  - Show the conditions are satisfied
- The Method
  - Assign a postcondition of True
  - Transform post conditions to generate preconditions
  - Total function has precondition of True

Details: Safe and Prec

- Safe(α) – the conditions for α to be safe
  - Safe(`s->tail`) = `s::cons`
- Prec(α, β) – the condition β, with α
  - Prec(`x = y`, `x::cons`) = `y::cons`
  - {`y::cons`} `x = y` {`x::cons`}

Flow Structures (if)

- $\{\alpha\}$ `if (cond) t; else f;` $\{\beta\}$
- α = safe(`cond`) $\wedge$

  (`cond` $\Rightarrow$ safe(`t`) $\wedge$ prec(`t`, β)) $\wedge$

  ($\neg$`cond` $\Rightarrow$ safe(`f`) $\wedge$ prec(`f`, β))

A small example

```
if (s::nil || s->head > i)
```

{True} `*s = *cons(i,copy(s));` {True}

{True}

- Now lets expand the `||` …

Expanding out the | |

{(s::nil ⇒ True) ∧ (¬s::nil ⇒ s::cons)}

```
if {True} (s::nil)
    {True} stmt;
else {s::cons} if {s::cons} (s->head > i)
    {True} stmt;
```

Equivalent to: {True}

Ingredients of Checking

- Prec and Safe functions
- A predicate solver
- Fixed pointing for loops
- Check that acyclic property is preserved
- Check all loops terminate

Back to the example

- The precondition to main is True
- The precondition to insert is True
- Both are total functions

- Also tested on Queues, Binary Trees, 234 Trees, for insertion and deletion
  - Proves all to be total functions

Future Work

- Use a mainstream language, i.e. C++
- Extend Pasta with static typing, arithmetic
- Operate on individual procedures
  - Currently it expands them ALL inline
- Make it go faster
  - Some runs took hours (`insert` in 234 Tree)
  - Profiling gave 20x speedup with ease

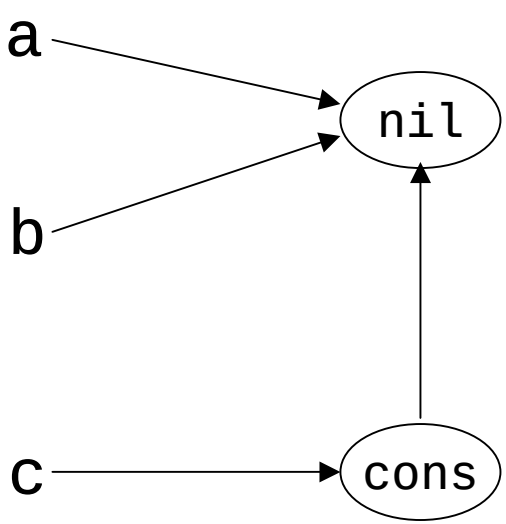
Total Pasta:

Unfailing Pointer Programs

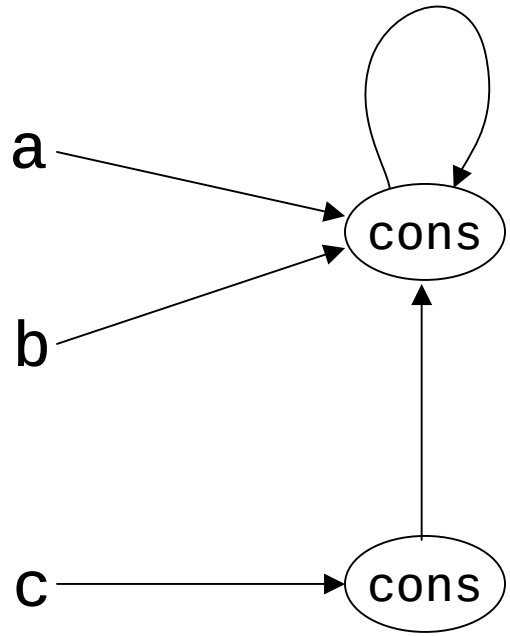
Neil Mitchell, ndm AT cs.york.ac.uk

Department of Computer Science, University of York

Starred Assignment




```
*a = *c
```



Notice that the value of b changes, without being mentioned