Research Languages

Peter C. Chapin
CIS-3030, Vermont Technical College
There is much ongoing PL research

- **Theory**
  - Describing concurrency
  - Advanced type systems
  - Interactions with the theory of computation

- **Practice**
  - Advanced garbage collection
  - Compilation and optimization strategies
  - Distributed and parallel programming
  - Meta programming
  - Small (and large) scale systems
Virgil

- **Virgil: Objects on the Head of a Pin**
  - By Ben L. Titzer given at OOPSLA 2006
- An OO language for very small systems

Problem:

- Microcontroller systems
  - Use interrupts extensive
  - Software must access low level hardware resources
  - Very small memories and CPU resources
- Convention OO language not suitable
  - Too resource intensive
  - Too abstract
Virgil Constraints

- No runtime system
  - Takes too many resources.
  - Thus no garbage collection
  - WYSIWYG

- No intrinsics
  - Aside from primitive types, no “built in” facilities

- No dynamic memory allocation
  - Requires run time support

- Minimal metadata
  - Such as virtual dispatch tables, etc.
Virgil is OO

- Class oriented
  - Like C++ or Java
- Single inheritance
- All methods are virtual (using a C++ term)
- Classes do not automatically inherit from Object
  - “Orphan” classes can be implemented with less overhead
  - Many classes in real programs are orphans.
Initialization vs Runtime

- Virgil distinguishes between program initialization and program execution.
  - Most languages do initialization as the first part of execution.
  - Virgil does initialization *during compilation*!

- Once initialization is complete, no objects can be created or destroyed.
  - Program is entirely static after it is compiled.
  - Very low runtime overhead in space in time.

- Compiler is complex
Example

- From the Virgil paper:

```java
class List {
    field head : Link;
    method add(i : int) { ... }
}

component K {
    field a : List = new List();
    field b : List;
    constructor() {
        b = new List();
        a.add(0);
        a.add(1);
        b.add(1);
    }
}
```
Complex Compiler

- Compiler executes the initialization
  - Runs the constructors
  - Does a garbage collection cycle to remove objects not actually needed by the final program
  - Writes out fully initialized objects into the ROM of the microcontroller

- All memory allocation is done before the program is deployed

- Compiler contains a full Virgil interpreter
  - Is Turning complete; compilation might run forever
A “dependent type” system for C

- Describes relationships between multiple data items
  - Example: A pointer to the start of an array together with the size of the array.
  - Example: A pointer into the middle of an array together with the bounds on the array.
  - Example: A union together with a tag value that distinguishes between the union members.

In conventional C the relationships above are not known to the compiler
- Thus the compiler can't check them
Example

• From the Deputy paper

```c
int sum
   (int * count(end-buf) buf, int *end)
{
   int sum = 0;
   while (buf < end) {
      sum += *buf;
      buf = buf + 1;
   }
   return sum;
}
```
Infer Missing Annotations

- Deputy compiler converts this to

```c
int sum
(int * count(end-buf) buf, int *end)
{
    int sum = 0;
    while (buf < end) {
        sum += *buf;
        int tmplen = (end - buf) - 1;
        int * count(tmplen) tmp = buf + 1;
        buf = tmp;
    }
    return sum;
}
```
Type Checking

- Deputy compiler then adds dynamic checks
  
  ```c
  int sum
  (int * count(end-buf) buf, int *end)
  {
  int sum = 0;
  while (buf < end) {
    assert(0 < end - buf);
    sum += *buf;
    int tmplen = (end - buf) - 1;
    assert(0 <= 1 <= end - buf);
    int * count(tmplen) tmp = buf + 1;
    assert(0 <= end - tmp <= tmplen);
    buf = tmp;
  }
  return sum;
  }
  ```
Finally...

- Compiler optimizes the run time checks
- Overall effect:
  - Original code will not violate the bounds on the array without it being noticed (by the compiler or at run time).
  - Allows for safer C programming without changing the run time data representation
    - In contrast other methods (e.g. Cyclone) involve changing the binary code produced by the compiler
Just a Sample

- Two research languages, Virgil and Deputy
  - Both targeting small scale systems
  - Virgil uses meta programming methods to reduce run time overhead
  - Deputy uses an advanced type system to add safety to C without adding any overhead (above normal C)

- Many, many other projects are out there!