A Type System for Dimensional Analysis
Pokey Rule and David Cheriton

pokey@stanford.edu, cheriton@cs.stanford.edu

**Typical Coding Errors**

```c
double distance; // Meters
double rate; // Meters / Second
double time; // Minutes
distance = rate; // Incorrect

double distanceFt; // Feet
double timeMins; // Minutes
distanceFt = rate * timeMins; // Incorrect
```

**Solution**

- A type system for representing physical measurements
- The user declares abstract dimension types for various measurable spaces:
  ```c
  Length : type : Dimension;
  Duration : type : Dimension;
  ```
- The user declares derived measurement types, indicating the underlying data type to be used and conversions between compatible data types:
  ```c
  Meters : type : Length, double {
  Feet : type : Length, double {
    = 0.3048 Meters;
  }
  
  Milliseconds : type : Duration, int {
  Seconds : type : Duration, double {
    = 1000 Milliseconds;
  }
  
  Minutes : type : Duration, double {
    = 60 Seconds;
  }
  ```
- When variables with measurement type are multiplied or divided, the type of the resulting object is the product or quotient, respectively, of the types of the multiplicands:
  ```c
  tripDist : Meters;
  tripRate : Meters / Seconds;
  tripTime : Seconds;
  tripDist = tripRate * tripTime;
  ```
- The type system ensures that the user does not try to add, subtract or assign incompatible measurement types:
  ```c
  tripDist = tripDist + tripRate; // Error
  tripDist = tripRate; // Error
  ```
- Unit conversion is automatic:
  ```c
  distFt : Feet;
timeMins : Minutes;
distFt = tripRate * timeMins;
  ```

**Advantages**

Fast: Generates C++ code as efficient as handwritten code using `double`, `int`, etc.
Safe: More errors are caught at compile time
Readable: Variables are labelled with their semantic meaning
Extensible: Defining new dimensions and measures is simple, and the compiler will automatically define product and quotient measure types, such as Luminens * Meters

**Generated Code**

```c
double tripDist;
double tripRate;
double tripTime;
tripDist = tripRate * tripTime;

double distFt;
double timeMins;
distFt = 60/0.3048 * tripRate * timeMins
```

**Implementation**

- Compiler has special Measure type node for representing compound types (Meters / Seconds, Meters / Luminens, etc) which contains a hashmap from typeName to unit exponent.
- For example, Meters / Seconds would be {Meters → 1, Seconds → -1}
- When the compiler parses an expression containing the product or quotient of two measure types, it creates a new Measure type node, adding or subtracting, respectively, the unit exponents of the multiplicands