THE SETUP PROGRAMMING LANGUAGE


December 23, 2011
Overview and Motivation
Setup is a language for manipulating sets and tuples

Overview
- Uses *C-style* syntax
- Additional *tuple* and *set* objects
- Elementary operations on sets are supported (e.g., union, intersection, set-difference)

Motivation
- The language of set theory is widely-used in mathematics and science
- Programs can be written quickly without verbose code
- Natural and intuitive way to think about large data sets
Objectives
In creating Setup, we had three objectives in mind

1. Create, modify and manipulate sets and tuples in a natural and intuitive way
2. Minimal code generation
3. Allow for both ordered and unordered collections of data
The Basics

Basic Elements of our Language

<table>
<thead>
<tr>
<th>Primatives</th>
<th>Objects</th>
<th>Set Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>set</td>
<td>union</td>
</tr>
<tr>
<td>string</td>
<td>tuple</td>
<td>intersect</td>
</tr>
<tr>
<td>double</td>
<td></td>
<td>cross</td>
</tr>
<tr>
<td>bool</td>
<td></td>
<td>minus</td>
</tr>
</tbody>
</table>

- The usual arithmetic operations +, -, *, / are provided.
- The + operator is overloaded for string concatenation.
Sets and Tuples

Tuples

\([(1,"a"),2) \ // type: \ ((int,str),int)\]
\([(1,2),3) \ // type: \ ((int,int),int)\]

Set Initialization

```plaintext
set A = \{4, 5, 6\};
set B = \{1 \ldots 6\};
set C = \{(x,y) \mid x \in A, y \in B\};
```
Example Functions

Functions

function main[] returns int { /* ... */}
function symDiff[set A, set B] returns set {
    return (A union B) minus (A intersect B);
}

- Execution begins with main
- Functions must be defined before they are called
- Function definition nesting is not permitted
Figure: Overview of Setup Compiler
Implementation

- Compiler can verify most type assignments by looking in a single frame of function context
- Setup supports a compact syntax we call the Set Builder expressions
- Set Builder has a lambda style syntax
- Variables are defined by the result of an expression evaluation
Lessons Learned

- Do not underestimate the task of building a compiler
- Just because the road is at first downhill, does not mean it does not lead to a cliff