PyPy -- where we are now

Michael Hudson mwh@python.net
Heinrich-Heine-Univeristät Düsseldorf

Introduction

- PyPy is:
  - an implementation of Python, written in Python
  - Aims for more flexibility and performance than existing implementations

Demo

- We can currently produce a binary that looks very much like CPython
- It's fairly slow (around the same speed as Jython)
- Starts pretty fast though :-)

Overview

- PyPy can be has two major components:
  - The "standard interpreter" which implements the eval loop and object semantics of CPython
  - The analysis tool chain that can analyse the standard interpreter and, for example, compile it to C.

The Standard Interpreter

- Written in RPython, but runs on CPython too
  - RPython is approximately defined as “what the analysis tool chain accepts”
- Consists of a parser/compiler, a bytecode interpreter and a Standard Object Space

The Interpreter/ Object Space split

- The byte code interpreter treats objects as black boxes and consistently references a "space" object to manipulate them
- This allows us to use a funky object space to help analysis (later)
- The Standard Object Space implements objects that look very much like CPython's.

The Parser/ Compiler

- The standard interpreter includes a parser and bytecode compiler for Python
- Based on work of Jonathan David Riehl and CPython’s compiler package
- Allows/will allow runtime modification of syntax and grammar

The Analysis Tool Chain

- Has four main parts:
  - The Flow Object Space
  - The Annotator
  - The RTyper
  - The Low Level backend

Flow Object Space

- Technically speaking, an “abstract domain” for the bytecode interpreter
- Treats objects as either “Constants” or “Variables”
- Works on a code object at a time
- Produces a control flow graph
- Basically stable since early 2005

The Annotator

- The RPythonAnnotator analyses an entire RPython program to infer types and inter-function control flow
- XXX more here
- More-or-less stable since early summer 2005

The RTyper

- First of all: “RTyper” is not a good name
- Converts the still-fairly-high-level output of the annotator into lower level operations that are easier to translate into languages like C.
- In particular, removes polymorphic calls from the graph.
- Basically working since summer 2005

The Low-Level Backend(s)

- Take the low-level operations produced by the RTyper and converts to a low-level language
- At time of writing, C and LLVM are the supported targets
- Working though not stable from spring 2005