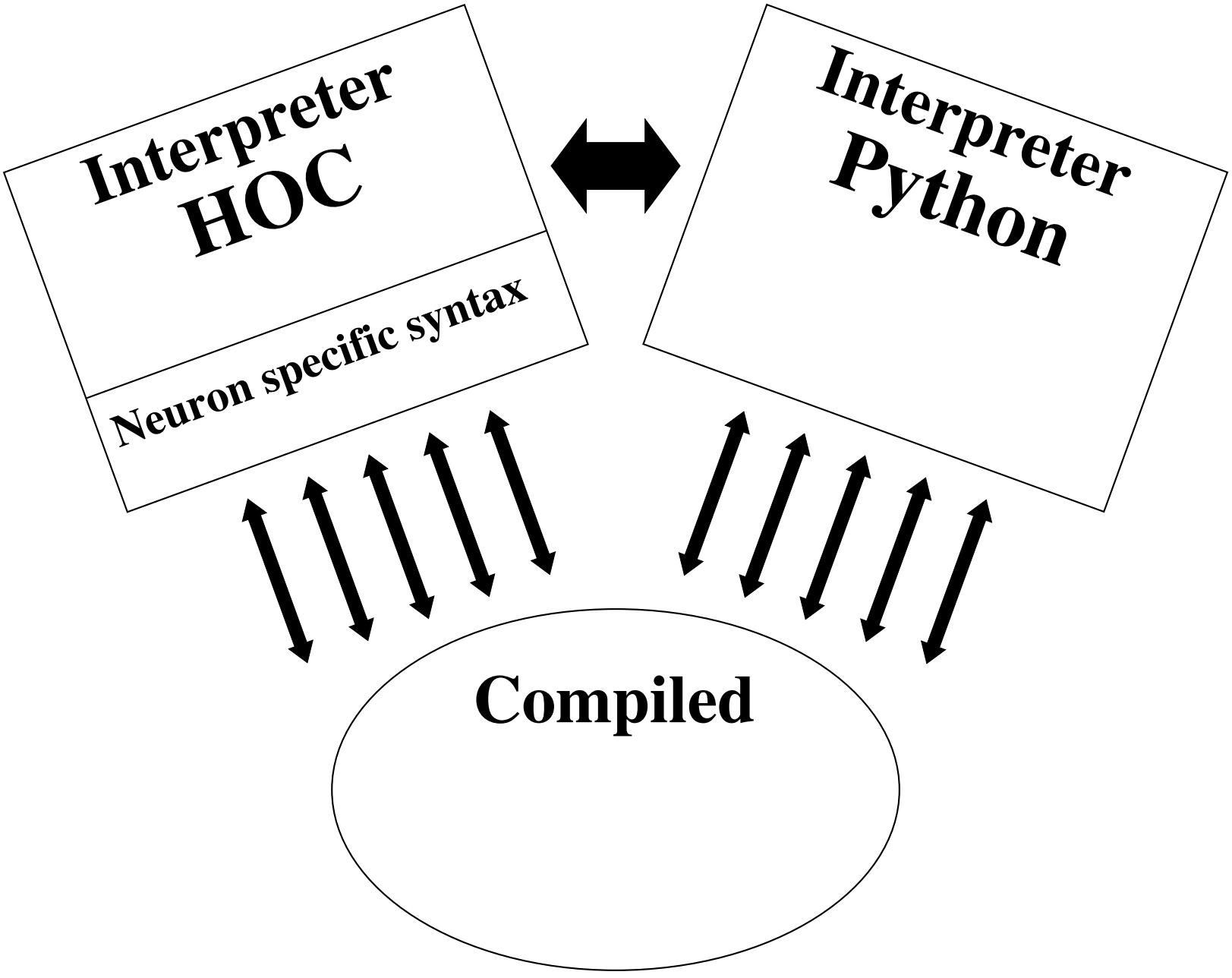
NEURON Python Interface Strategies

Michael Hines

CodeJam#10
Heidelberg 2019

NINDS


Interpreter
HOC
Neuron specific syntax
Interpreter
Python
Compiled

Hoc Vector

```
from neuron import h
v = h.Vector(range(5))
v.printf()
    0    1    2    3    4
```

Hoc Vector

```
from neuron import h
v = h.Vector(range(5))
v.printf()
    0    1    2    3    4
```

Numpy data sharing

```
npv = v.as_numpy()
print (type(npv))
    <class 'numpy.ndarray'>
npv[3] = 42
print (v[3])
    42.0
```

Hoc Vector

Pickling

```
$ mpiexec –n 2 python3 –c ’
from neuron import h
h.nrnmpi_init()
pc = h.ParallelContext()

v = h.Vector(range(5)) if pc.id() == 0 else None
vv = pc.py_broadcast(v, 0)
if pc.id() == 1:
  vv.printf()

pc.barrier()
h.quit()
’
```

```
    numprocs=2
    0    1    2    3    4
```

Hoc Vector, Numpy share data

```
Object** vec_as_numpy_helper(int size, double* data) {
  if (vec_as_numpy) {
    PyObject* po = (*vec_as_numpy)(size, data);
```

Hoc Vector, Numpy share data

```
Object** vec_as_numpy_helper(int size, double* data) {
  if (vec_as_numpy) {
    PyObject* po = (*vec_as_numpy)(size, data);

static PyObject* (*vec_as_numpy)(int, double*);
int nrnpy_set_vec_as_numpy(PyObject* (*p)(int, double*)) {
  vec_as_numpy = p;
  return 0;
}
```

Hoc Vector, Numpy share data

```
int nrnpy_set_vec_as_numpy(PyObject* (*p)(int, double*)) {
  vec_as_numpy = p;
  return 0;
}


try:
  vec_to_numpy_prototype = ctypes.CFUNCTYPE(
    ctypes.py_object, ctypes.c_int,
    ctypes.POINTER(ctypes.c_double))
  def vec2numpy(size, data):
    try:
      return numpy_from_pointer(data, size)
    except:
  vec_to_numpy_callback = vec_to_numpy_prototype(vec2numpy)
  set_vec_as_numpy = nrn_dll_sym('nrnpy_set_vec_as_numpy')
  set_vec_as_numpy(vec_to_numpy_callback)
```

Pickle Hoc Vector

```
static PyMethodDef hocobj_methods[] = {
    ...
{"__reduce__", hocpickle_reduce, METH_VARARGS, "pickle interface"},
{"__setstate__", hocpickle_setstate, METH_VARARGS, "pickle interface"},
{NULL, NULL, 0, NULL}};
```

Pickle Hoc Vector

```
static PyObject* hocpickle_reduce(PyObject* self, PyObject* args) {
    PyObject* mod = PyImport_ImportModule("neuron");
    PyObject* pkl = PyObject_GetAttrString(mod, "_pkl");
    PyObject* ret = PyTuple_New(3);
    PyObject* state = PyTuple_New(4);

    return ret;
}

def _pkl(arg):
  return h.Vector(0)
```

Pickle Hoc Vector

```
static PyObject* hocpickle_reduce(PyObject* self, PyObject* args) {
    PyObject* mod = PyImport_ImportModule("neuron");
    PyObject* pkl = PyObject_GetAttrString(mod, "_pkl");
    PyObject* ret = PyTuple_New(3);
    PyObject* state = PyTuple_New(4);

    PyTuple_SET_ITEM(ret, 0, pkl);
    PyTuple_SET_ITEM(ret, 2, state);

    return ret;
}

def _pkl(arg):
  return h.Vector(0)
```

Pickle Hoc Vector

```
static PyObject* hocpickle_reduce(PyObject* self, PyObject* args) {
    PyObject* mod = PyImport_ImportModule("neuron");
    PyObject* pkl = PyObject_GetAttrString(mod, "_pkl");
    PyObject* ret = PyTuple_New(3);
    PyObject* state = PyTuple_New(4);

    PyTuple_SET_ITEM(ret, 0, pkl);
    PyTuple_SET_ITEM(ret, 2, state);

    Vect* vec = (Vect*)((PyHocObject*)self)–>ho_–>u.this_pointer;
    Pyobject* str = PyBytes_FromStringAndSize((const char*)
        vector_vec(vec), vec–>capacity() * sizeof(double));
    PyTuple_SET_ITEM(state, 2, PyInt_FromLong(vec–>capacity()));
    PyTuple_SET_ITEM(state, 3, str);

    return ret;
}

def _pkl(arg):
  return h.Vector(0)
```

Binary distribution generality.

```
cmake .. –DNRN_ENABLE_PYTHON_DYNAMIC=ON
    –DNRN_PYTHON_DYNAMIC=’python3.6;python3.7;python3.8’
make –j install
```

Binary distribution generality.

```
cmake .. –DNRN_ENABLE_PYTHON_DYNAMIC=ON
    –DNRN_PYTHON_DYNAMIC=’python3.6;python3.7;python3.8’
make –j install
```

lib/python/neuron

- hoc.cpython–36m–darwin.so
- hoc.cpython–37m–darwin.so
- hoc.cpython–38–darwin.so

Binary distribution generality.

cmake .. –DNRN_ENABLE_PYTHON_DYNAMIC=ON
–DNRN_PYTHON_DYNAMIC=’python3.6;python3.7;python3.8’
make –j install

lib/python/neuron
hoc.cpython–36m–darwin.so
hoc.cpython–37m–darwin.so
hoc.cpython–38–darwin.so

Since Python3.2, can use a stable ABI if
one uses only a subset of the API.
#define PY_LIMITED_API
cmake arg –DNRN_PYTHON_ABI3=ON

hoc.abi3.so

Binary distribution generality.

lib/libnrnpython3.dylib

```
static PyType_Slot nrnpy_HocObjectType_slots[] = {
    ...
    {Py_tp_call, (void*)hocobj_call},
    {Py_tp_getattro, (void*)hocobj_getattro},
    {Py_tp_setattro, (void*)hocobj_setattro},
    {Py_tp_richcompare, (void*)hocobj_richcmp},
    ...
    {0, 0},
};
```

Binary distribution generality.

Unfortunately, cython generated files...

.../python3.7/importlib/_bootstrap.py:219:
RuntimeWarning:
compiletime version 3.6 of module
'neuron.rxd.geometry3d.ctng'
does not match runtime version 3.7