LV2 Atoms: A Data Model for Real-Time Audio Plugins

David Robillard

School of Computer Science
Carleton University

May 1, 2014
What Not to Do

(or: Why Extensibility Matters)
LV2 Events

- LADSPA had only float I/O
- LV2 added generic events:
  - time
  - size
  - type
  - body...
- Type is a URID: a URI mapped to an integer
- Truly extensible (anyone can define event types)
Event Types in Practice

- MIDI for notes, etc.
- ... and that’s about it
- Yet many developers need more power
What to do?

- What message format(s) to use?
- MIDI? OSC? text (JSON, Turtle, sexps)?, blobs? ...
Idea

- Some good message options
- ... but what about structured data?
- Can we simplify/genericize what we already have?
- Hmm... event − time = ?

<table>
<thead>
<tr>
<th>time</th>
<th>size</th>
<th>type</th>
<th>body...</th>
</tr>
</thead>
</table>
Atom: A container for anything

typedef struct {
    uint32_t size;
    uint32_t type;
} LV2_Atom;

- A simple-as-possible “universal” idea (size aside)
- **Win:** this is all you need to know to copy atoms
- Hosts, routers, etc., can simply memcpy
Building the Foundation

- Opacity is nice, but “meaningless” blobs are problematic
- What can we build on this ground?
- Primitives are easy, e.g. the body of an Int is just that:

  ```c
  typedef struct {
    LV2_Atom atom;
    int32_t body;
  } LV2_Atom_Int;
  ```

- Wonderful: ints, floats, strings, etc.
Structured Data in the Ivory Tower

- To build larger structures, we need **collections**
- A pure and simple model: lists + dictionaries (ala JSON)
  - **Tuple**: atom, atom, atom, ...
  - **Object**: key, atom, key, atom, ...

David Robillard

LV2 Atoms: A Data Model for Real-Time Audio Plugins
Structured Data in the Noisy Trenches

- For block-processed audio: time stamps
  - Sequence: time, atom, time, atom, ...

- For high performance / SIMD: vectors
  - Vector: atom body, atom body, ...
  - (Like Tuple but homogeneous with headerless elements)

- For LV2: Turtle compatibility
  - Object keys are URIDs (we’ll see why shortly)
Object as Message

- We can “think in Turtle”, though messages are binary
- ... and/or convert between the two with code (serialization)
- ... and/or write atoms in plugin data files

```plaintext
eg:control
  lv2:minimum 0.0 ;
  lv2:maximum 1.0 ;
  lv2:default 0.5 .
```
Time: Position and Speed

```
a time:Position ;
time:frame 88200 ;
time:speed 0.0 ;
time:bar 1 ;
time:barBeat 0.0 ;
time:beatUnit 4 ;
time:beatsPerBar 4.0 ;
time:beatsPerMinute 120.0 .
```
Loading Samples

```
[]

a patch:Set ;
patch:property eg:sample ;
patch:value </media/bonk.wav> .
```
Where to go from here?

- Lots of potential without adding new APIs
- Event-based parameter control?
- Max-like programming-with-plugins?
- What else can we make plugins do?