An Introduction to SWI-DSP

July 24, 2008

1 What is

SWI-DSP is a bunch of modules implemented in SWI-Prolog that defines a unified and declarative framework to work with DSP resources. It allows a user to create complex DSP workflows and execute them through Prolog queries.

As part of a larger distributed architecture for multimedia knowledge management (often called "Knowledge Machine") it can interact with other modules that provide more data sources, data management, access interfaces... this is the case of Henry which links this DSP engine to Semantic Web technologies.

2 Who wrote it

David Pastor Escuredo. Collaborators: Yves Raimond, Samer Abdallah

3 Why Prolog

It may seem weird to use Prolog as basis for an such engine. However, Prolog features interesting capabilities that make it suitable for our purposes. As a powerful query language close to 1st order logic, Prolog can be used to express relational data (M:N 1:M relationships) as well as a natura way to deal with logic allowing recursive rules, transitive relationships...

Moreover, Prolog predicates can be used to wrap computations as functional predicates triggering heavy data computations and unifying these predicates with other types of data and relationships.

4 Components

SWI-DSP is formed of several submodules wrapping external APIs and libraries which perform the actual computations. Thus, SWI-DSP is enriched by the nicer development interfaces of such APIs and the algorithms implemented in them.

- swiaudio: a module for audio decoding and signals information. It wraps several open source libraries: MAD, FISHSOUND, OGGZ, SOUNDFILE and FAAD.

- swivamp: this module defines a VAMP Plugins host which allows us to drive and query the plugins in our system.
- swiladspa: this module defines the same functionality for LADSPA plugins
- swiweka: this a bit off-topic module that easily contributes to MIR tasks. It defines a machine learning module driven from Prolog that interfaces the WEKA API.

Modules are layered in different levels of abstraction from API communication predicates to declarative predicates which have semantics closer to a DSP researcher understanding.

5 How to get it and run it

The related components which installation is required are: ladspa sdk, vamp sdk, mad lib, fishsound lib, oggz lib, sndfile lib, faad lib, qt4 library, weka, and SWI-Prolog.

The project (which also includes Henry and other modules) is available at `http://code.google.com/p/km-rdf/`. After that, the project can be built with a general Makefile that needs some local settings.

Specific instructions for every module are in each README file. Each module can be loaded from the Prolog prompt or all at the same time from swi-dsp or henry folders. e.g.

```
?-use_module('vamp_transform').
```

6 Modules interaction

Each module enhances each other and collaborate to create complex DSP workflows. For example we can use swiaudio to decode files from our collection, then apply some processing using ladspa, features can be extracted using vamp and finally we can use weka to build some classifier using features as parameters.

This communication is possible thanks to common types (Prolog functors representing common DSP objects) and a common format for large binary data (BLOBS) as identifiers which are indirect and universal references to data objects.

7 SWI-DSP and other software

This module can interact with any other SWI-Prolog module. Henry provides persistency, declarative data transactions and an N3/Prolog entailment module that allow us to drive computations by SPARQL queries and refer to them by URIs within a semantic namespace defined in an OWL ontology.

However this module can be loaded by other reasoners (Samer's system) or from external hosts that may use this engine to display or store DSP processes.

8 Status

The system as presented is quite steady and complete. Other wrappers can be implemented enriching the vocabulary of SWI-DSP.

It works for Linux and MAC OS/X (but may be a bit more difficult to compile).

There are some packages that can be download as release 1.0 at the google code page.

9 Other documents

- Wiki pages at `http://code.google.com/p/km-rdf/`
- README files
- DBtune site
- Paper waiting for acceptance