A format for virtual orchestras: FlowML

Version 0.5 – DRAFT*

Bert Schiettecatte (57986)
bschiett@vub.ac.be

May 2, 2000

Abstract

This document proposes a format for storing synthesis diagrams and their supporting mechanisms. Such a standard format is necessary to allow exchange of virtual musical instruments between several (non-)realtime software synthesizers, and to publish synthesis diagrams on the WWW.

*Software designers should not rely on this document until the first major version.

Contents

1 Introduction

This document proposes a standard format for synthesis flow charts. The format will be defined later in this document as a collection of XML DTDs. The primary motivation for such an exchangeable format was the problem of serializing collections of reusable signal processing constructs as found in (non-)realtime software synthesizers.

2 Definitions

To be completed

3 To chart or not to chart

When the idea for this universal format first arised, it was obvious that there should be some primitive structures defined for the format, such that compilers from FlowML to some target synthesis language would be able to generate code for these primitive structures.

The first idea was to introduce a set of primitives from synthesis languages like SAOL and CSound: various mathematical operators, along with signal processing operators. It turned out that this is (probably) not the right solution: these are constructs from a (more or less) sequential language, and a diagram does not specify a real order on its components, the way a programming language does (there was the assumption for a while that a synthesis algorithm could be drawn very easily as a directed graph). Therefore, a collection of primitive synthesis *building blocks* will be introduced in this document, and tools who want to support the FlowML format must provide support for these blocks.

After discussion with several people from the `saol-dev` mailing-list and various members of the university where this project is supervised, it would be advisable to introduce a mechanism in the format which allows for target-language specific building blocks (blocks which cannot be defined using the primitive blocks introduced in this document). At this time such a mechanism isn't included yet, but it is planned for the next version of this document.

4 Fundamental concepts

The format supports two types of signals: *audio* and *control*. These signals are not interchangeable and cannot be converted from one to another. All signals are mono: if a stereo signal is required, two mono outputs should be used, instead of one stereo signal.

5 Primitive structures

Based on some popular realtime software synthesizers, a few primitive building blocks will now be presented. The list is by no means complete at this time, and most descriptions are missing.

5.1 Inputs

1. Audio in
2. Constant in
3. Controller in

5.2 Outputs

1. Audio out
2. Controller out

5.3 Logic

1. Switch
2. And
3. Or
4. Not

5.4 Arithmetic operators

1. Add
   Takes two control signals and simply adds them.
2. Multiply
3. Substract
4. Divide

5.5 Amplifiers, mixers & envelope generators

1. Crossfade
2. Pan
3. Amplify
4. Mix
   Takes two audio signals and mixes them.
5. ADSR

5.6 Oscillators

1. Sawtooth
2. Pulse
3. Sine

5.7 Discrete oscillators

1. LFO
   Takes two control signals, *frequency* and *variation.* It has one control output.

5.8 Filters

1. High-pass 1-pole
2. Low-pass 1-pole

5.9 Effects

1. Delay

   Takes an audio signal (the dry signal) and a control signal *time*, and outputs an audio signal (the delayed version of the input signal).

2. Reverb

3. Spatialize

6 Examples

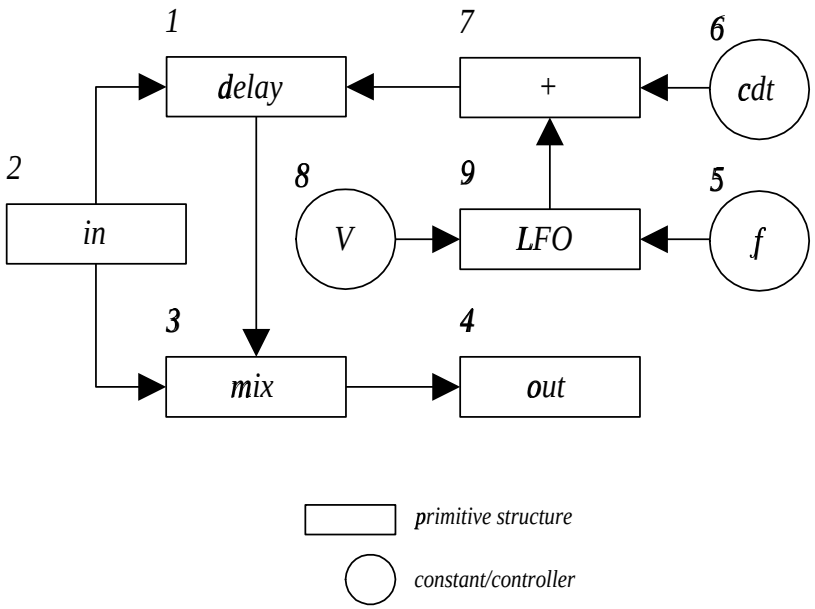
6.1 Orchestra definition

An example orchestra:

```
<?xml version="1.0"?>

<!--

    FlowML Orchestra

-->

<!DOCTYPE Orchestra SYSTEM "Orchestra.dtd">

<Orchestra>
        <Instruments>
                <InstrumentInstance id="flanger" label="Flanger">
                        <Route target_id="output_bus"/>
                </InstrumentInstance>
                <InstrumentInstance id="classic_synth" label="Classic Synth">
                        <Route target_id="dry_bus"/>
                </InstrumentInstance>
        </Instruments>
        <Buses>
                <Bus id="dry_bus" label="Dry instruments" width="1">
                        <Send target_id="flanger"/>
                </Bus>
                <Bus id="output_bus" label="Output bus" width="2" output="yes">
                </Bus>
        </Buses>
</Orchestra>
```

6.2 Instrument definition

The following is a modified version of the flanger circuit from [Roads96], p. 438. The diagram was redesigned in terms of structures from 5:



The corresponding FlowML document would be:

```
<?xml version="1.0"?>

<!--

    FlowML Flanger instrument

-->

<!DOCTYPE Instrument SYSTEM "Instrument.dtd">

<Instrument id="flanger">
    <Description>
        A classic flanger effect.
    </Description>
    <Implementation>
        <StructureInstance id="delay" label="delay" type="delay">
            <Connection from="audio_out" to="in1" target_id="mix"/>
        </StructureInstance>
        <StructureInstance id="in" label="input" type="audio_input">
            <Connection from="out" to="audio_in" target_id="delay"/>
            <Connection from="out" to="in2" target_id="mix"/>
        </StructureInstance>
        <StructureInstance id="mix" label="mix" type="mixer">
            <Connection from="out" to="in" target_id="out"/>
        </StructureInstance>
        <StructureInstance id="out" label="output" type="audio_output">
        </StructureInstance>
        <StructureInstance id="f" label="frequency" type="constant">
            <Connection from="out" to="frequency_in" target_id="LFO"/>
        </StructureInstance>
        <StructureInstance id="cdt" label="central time" type="constant">
```

```
            <Connection from="out" to="in1" target_id="plus"/>
        </StructureInstance>
        <StructureInstance id="plus" label="add" type="addition">
            <Connection from="out" to="time_in" target_id="delay"/>
        </StructureInstance>
        <StructureInstance id="V" label="Variance" type="constant">
            <Connection from="out" to="variation_in" target_id="LFO"/>
        </StructureInstance>
        <StructureInstance id="LFO" label="LFO" type="lfo">
            <Connection from="out" to="in2" target_id="plus"/>
        </StructureInstance>
    </Implementation>
</Instrument>
```

6.3 Structure definition

An example structure definition:

```
<?xml version="1.0"?>

<!--

    FlowML Structure

-->

<!DOCTYPE Structure SYSTEM "Structure.dtd">

<Structure id="flanger">
    <Description>
        A classic flanger effect.
    </Description>
    <Implementation>
        <PrimitiveInstance id="delay" label="delay" type="delay">
            <Connection from="audio_out" to="in1" target_id="mix"/>
        </PrimitiveInstance>
        <PrimitiveInstance id="in" label="input" type="audio_input">
            <Connection from="out" to="audio_in" target_id="delay"/>
            <Connection from="out" to="in2" target_id="mix"/>
        </PrimitiveInstance>
        <PrimitiveInstance id="mix" label="mix" type="mixer">
            <Connection from="out" to="in" target_id="out"/>
        </PrimitiveInstance>
        <PrimitiveInstance id="out" label="output" type="audio_output">
        </PrimitiveInstance>
        <PrimitiveInstance id="f" label="frequency" type="constant">
            <Connection from="out" to="frequency_in" target_id="LFO"/>
        </PrimitiveInstance>
        <PrimitiveInstance id="cdt" label="central time" type="constant">
            <Connection from="out" to="in1" target_id="plus"/>
        </PrimitiveInstance>
        <PrimitiveInstance id="plus" label="add" type="addition">
            <Connection from="out" to="time_in" target_id="delay"/>
```

```
        </PrimitiveInstance>
        <PrimitiveInstance id="V" label="Variance" type="constant">
            <Connection from="out" to="variation_in" target_id="LFO"/>
        </PrimitiveInstance>
        <StructureInstance id="LFO" label="LFO" type="lfo">
            <Connection from="out" to="in2" target_id="plus"/>
        </StructureInstance>
    </Implementation>
</Structure>
```

---

7 Document Type Descriptions

Below are the formal specifications for the format.

7.1 Orchestra

```
<!--

    FlowML Orchestra DTD

-->

<!ELEMENT Orchestra (Instruments, Buses)>

<!ELEMENT Instruments (InstrumentInstance)+>

<!ELEMENT Buses (Bus)+>

<!ELEMENT InstrumentInstance (Route)+>
<!ATTLIST InstrumentInstance
 id ID #REQUIRED
 label CDATA #REQUIRED>

<!ELEMENT Route EMPTY>
<!ATTLIST Route
 target_id IDREF #REQUIRED>

<!ELEMENT Bus (Send)*>
<!ATTLIST Bus
 id ID #REQUIRED
 label CDATA #REQUIRED
 width CDATA #REQUIRED
 output (yes|no) "no">

<!ELEMENT Send EMPTY>
<!ATTLIST Send
 target_id IDREF #REQUIRED>
```

7.2 Instrument

```
<!--
```

```
    FlowML Instrument DTD

-->

<!ELEMENT Instrument (Description, Implementation)>
<!ATTLIST Instrument id ID #REQUIRED>

<!ELEMENT Description (#PCDATA)>

<!ELEMENT Implementation (StructureInstance)+>

<!ELEMENT StructureInstance (Connection)*>
<!ATTLIST StructureInstance
 id ID #REQUIRED
 label CDATA #REQUIRED
 type CDATA #REQUIRED>

<!ELEMENT Connection EMPTY>
<!ATTLIST Connection
 from CDATA #REQUIRED
 to CDATA #REQUIRED
 target_id IDREF #REQUIRED>
```

7.3 Structure

```
<!--

    FlowML Structure DTD

-->

<!ELEMENT Structure (Description, Implementation)>
<!ATTLIST Structure id ID #REQUIRED>

<!ELEMENT Description (#PCDATA)>

<!ELEMENT Implementation (StructureInstance|PrimitiveInstance)+>

<!ELEMENT StructureInstance (Connection)*>
<!ATTLIST StructureInstance
 id ID #REQUIRED
 label CDATA #REQUIRED
 type CDATA #REQUIRED>

<!ELEMENT PrimitiveInstance (Connection)*>
<!ATTLIST PrimitiveInstance
 id ID #REQUIRED
 label CDATA #REQUIRED
 type CDATA #REQUIRED>

<!ELEMENT Connection EMPTY>
<!ATTLIST Connection
 from CDATA #REQUIRED
 to CDATA #REQUIRED
```

```
target_id IDREF #REQUIRED>
```

8 Acknowledgements

To be completed

References

[Roads96] C. Roads. *The Computer Music Tutorial.* 1996, MIT Press.