International Conference on New Interfaces for Musical Expression

TSL Synthesis Synthesizer: Reconfigurable Signal Flows through Program Synthesis

Michel Vazirani, Wonhyuk Choi, Mark Santolucito

TSL Synthesis Synthesizer: Reconfigurable Signal Flows through Program Synthesis

Michel Vazirani (Columbia University), Wonhyuk Choi (Columbia University), Mark Santolucito (Barnard College, Columbia University)

1. PubPub Link

https://nime.pubpub.org/pub/aa27fggy/draft?access=wn9qmyy9

2.Conference Abstraction

We introduce the TSL Synthesis Synthesizer, an online keyboard synthesizer that uses program synthesis to offer dynamic control over its own sound parameters. Program synthesis is a field of computer science aimed at automatically generating code that satisfies a given behavioral specification. The TSL Synthesis Synthesizer presents a novel approach at harnessing the expressivity of Temporal Stream Logic (TSL)[1] to create such specifications in order to dictate how sound parameters should be altered over time.

Fundamentally, the web page is a keyboard synthesizer. The keyboard can be played via one of three methods. Users can either click the keys on the webpage with their mouse, use their computer keyboard, or connect a USB MIDI keyboard. At a high level, the tool functions by first prompting the user to create a TSL specification describing how the sound parameters should be altered during performance depending on what note the user is playing on the synthesizer. Once a specification is defined, program synthesis is used to generate JavaScript code that manages the dynamic changes in the sound parameters. Then, this JavaScript code is embedded into the webpage, allowing users to play the keyboard and rely on the web page to automatically alter the sound parameters according to the specification.

From a technical perspective, the key contribution of this work is the use of program synthesis to automatically generate code based on users' specifications. Temporal Stream Logic is a logic for describing reactive systems - systems that infinitely consume input and produce output over time. In our case, our reactive system consumes user input (i.e. MIDI input) and produces output as signal flow rerouting (e.g. toggling an LFO). TSL formulae operate on an abstract notion of "time," which moves forward with each reactive input; in our example, each new MIDI signal moves time a step forward. TSL includes temporal operators such as "next" [◯], "always" [□

], “eventually” [⋄], “until” [U], “weak until” [W], “release” [R], and “as soon as”[A]. It also includes boolean logic operators such as “and”[∧], “or” [∨], “implies”[→], and “if and only if” [↔].

Using TSL, we can define the desired signal flow for composition shown in the video in Sec. 3:

(□ play note67 ↔ (◯[am ← toggle am])) ∧
(□ play note64 ↔ (◯[lfo ← toggle lfo]))

We can also create more complex control formulae, such as the following:

□ play note60 → ◯(□[am ← toggle am]) W play note67

Although our current web demo demonstrates the expressivity of TSL, it is limited in its practical applications. We intend to explore possibilities to determine the most applicable musical environments for TSL control, as well as options for using the TSL synthesis to control third-party audio tools (e.g. VST plugins).

3.MEDIA

Synthesizer: https://tslsynthesissynthesizer.com/

Visit the web version of this article to view interactive content.

TSL Synthesis Synthesizer Demo

—

Citations

1. Finkbeiner, B., Klein, F., Piskac, R., & Santolucito, M. (2019). Temporal stream logic: Synthesis beyond the bools. In *International Conference on Computer Aided Verification* (pp. 609–629). ↩