The T-calculus : Towards a structured programming of (musical) time and space.

David Janin et al., LaBRI, Université de Bordeaux
FARM 2013, Boston

1. An example

The bebop problem and the bebop solution [1]...

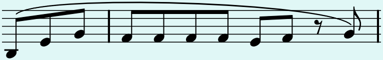
My little blue suede shoes (Ch. Parker)

Musical analysis

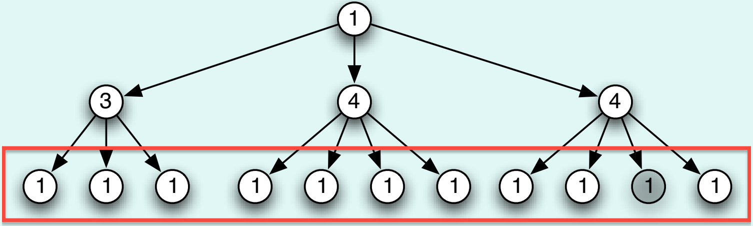
Three times motive (a) followed by its conclusive variant (b).

Play

String modeling (a)

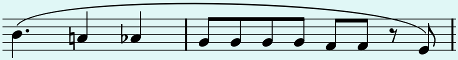


modeled by:

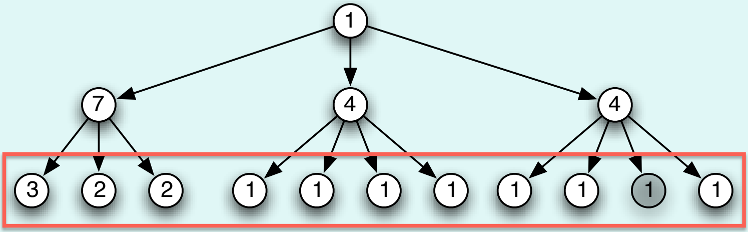


(a)

String modeling (b)

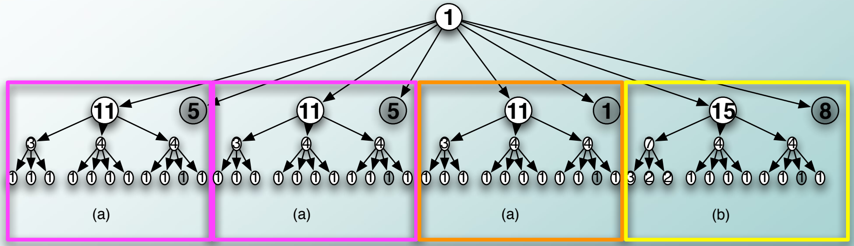


modeled by:



(b)

Resulting modeling

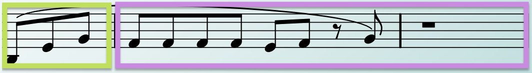


Problem:

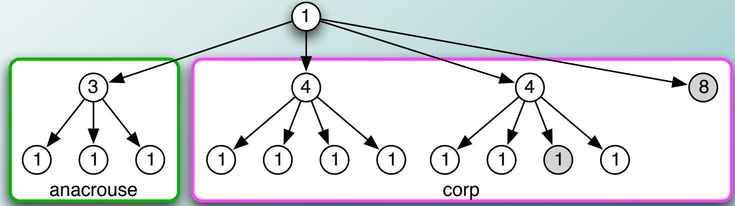
- we have inserted rests of various size : 5, 5, 1 and 8,
- we have lost the logical structure (3x(a) + (b)),
- handling variations will be even more messy.

Alternative : make the anacrusis and synchronization point explicit

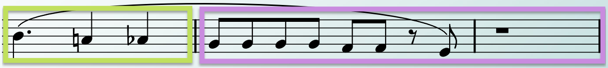
The "real" first pattern:



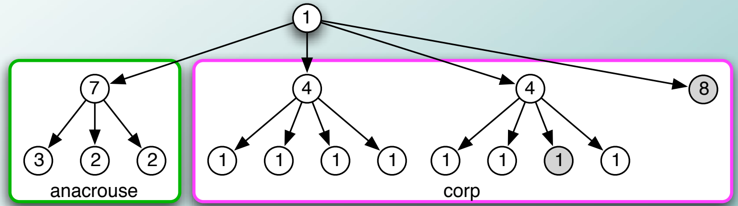
modeled by:



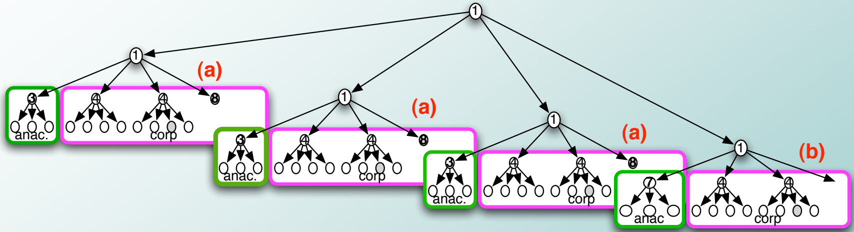
The “real” second pattern:



which give:



with resulting *mixed composition*:



defined with both sequential and parallel features.

Here comes back the logical structure: 3x(a) + (b) !

This is tiled strings (or streams) modeling !

2. Tiled streams

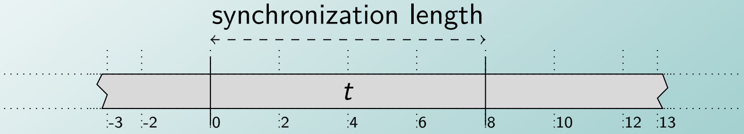
Embedding (audio) strings and (audio) streams into tiled streams [2]

Tiled streams

Basic types A, B, ..., extended with a special silence value 0.

Tiled stream

A *"bi-infinite" sequence* of values $t : \mathbb{Z} \to A$ with an additional *synchronization length* $d(t) \in \mathbb{N}$.

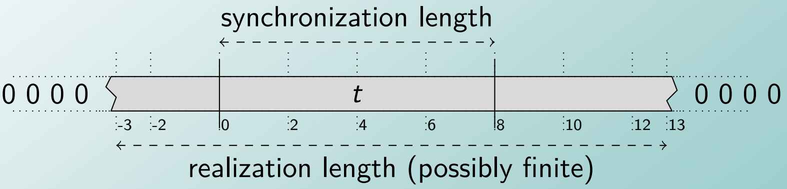


Tiled streams

Basic types A, B, ..., extended with a special silence value 0.

Tiled stream

A *"bi-infinite" sequence* of values $t : \mathbb{Z} \to A$ with
an additional *synchronization length* $d(t) \in \mathbb{N}$.

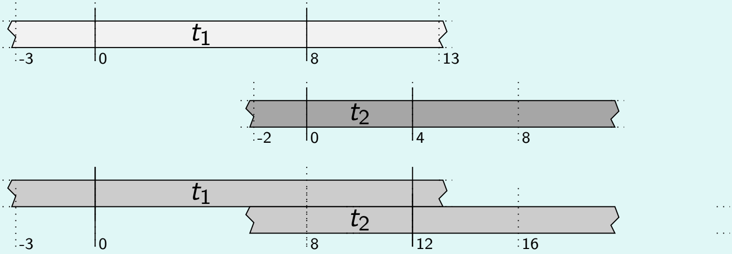


Tiled stream product: the “free” case

Tiled stream product

Two tiled stream $t_1 : \mathbb{Z} \to A$ and $t_2 : \mathbb{Z} \to B$ and their product $t_1; t_2 : \mathbb{Z} \to A \times B$ defined, for every $k \in \mathbb{Z}$, by

$$(t_1; t_2)(k) = (t_1(k), t_2(k - d(t_1)))$$



with resulting synchronization length

$$d(t_1; t_2) = d(t_1) + d(t_2)$$

Tiled stream product: the parameterized case

Let $op : A \times B \to C$

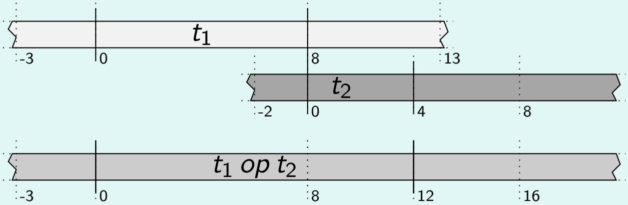
Operator lifting

Two tiled streams $t_1 : \mathbb{Z} \to A$ and $t_2 : \mathbb{Z} \to B$ let $t_1 \, op \, t_2 : \mathbb{Z} \to C$ defined by $d(t_1 \, op \, t_2) = d(t_1) + d(t_2)$ and

$$(t_1 \, op \, t_2)(k) = t_1(k) \, op \, t_2(k - d(t_1))$$

for every $k \in \mathbb{Z}$.

Synchronization + Fusion : $t_1 \, op \, t_2$



Tiled stream product: the parameterized case

Let $op : A \times B \to C$

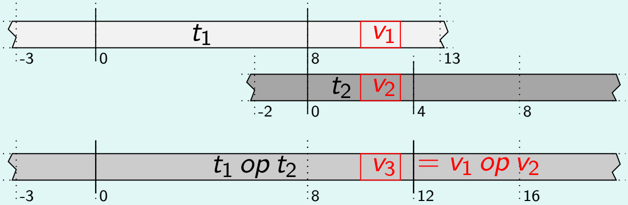
Operator lifting

Two tiled streams $t_1 : \mathbb{Z} \to A$ and $t_2 : \mathbb{Z} \to B$ let $t_1 \; op \; t_2 : \mathbb{Z} \to C$ defined by $d(t_1 \; op \; t_2) = d(t_1) + d(t_2)$ and

$$(t_1 \; op \; t_2)(k) = t_1(k) \; op \; t_2(k - d(t_1))$$

for every $k \in \mathbb{Z}$.

Synchronization + Fusion : $t_1 \; op \; t_2$



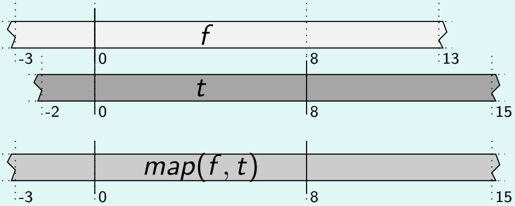
Tiled stream product: the "map" example

Let $map : (A \to B) \times A \to B$ defined by $map(x, y) = x(y)$

Tiled map

Two tiled streams $f : \mathbb{Z} \to (A \to B)$ and $t : \mathbb{Z} \to A$ with $d(f) = 0$ consider $map(f, t) : \mathbb{Z} \to B$.

$map(f, t)$



3. Re-synchronization

Reset and co-reset for resynchronization

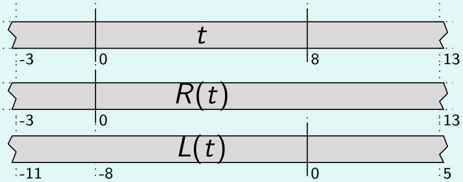
Natural operators: resynchronization

Left and right resynchronization

A tiled stream $t : \mathbb{Z} \to A$, the synchronization reset $R(t)$ and the synchronization co-reset $L(t)$ of the tiled stream t defined, for every $k \in \mathbb{Z}$, by

$$R(t)(k) = t(k) \text{ and } L(t)(k) = t(k - d(t))$$

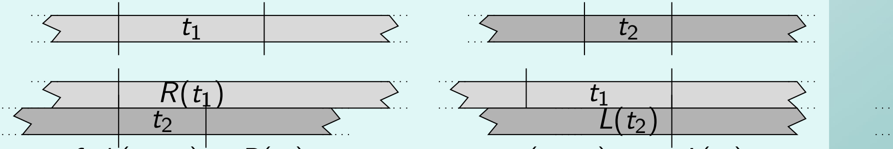
with synchronization length $d(R(t)) = d(L(t)) = 0$.



Derived operators: fork and join

Parallel fork and join

Tiled product and resets allows for defining parallel products:



$fork(t_1, t_2) = R(t_1); t_2$ $\quad$ $join(t_1, t_2) = t_1; L(t_2)$

with synchronization length $d(fork(t_1, t_2)) = d(t_2)$ and $d(join(t_1, t_2)) = d(t_1)$.

4. Embeddings

Links with Hudak's *Polymorphic Temporal Media* [3] and ω-semigroups [4]

Embedding strings and concatenation

Finite strings and concatenation

Two A-strings $u : [0, d_u[\to A$ and $v : [0, d_v[\to A$ and their concatenation

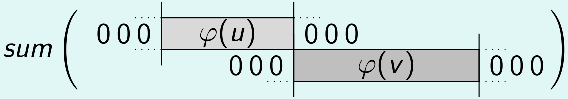
$$u \cdot v : [0, d_u + d_v[\to A$$

defined by

$$u \cdot v(k) = \begin{cases} u(k) & \text{when } 0 \leq k < d_u, \\ v(k - d_u) & \text{when } d_u \leq k < d_u + d_v. \end{cases}$$

$$\varphi : \mathit{strings} \to \mathit{tiledStreams}$$

String embedding: $\varphi(u) \cdot \varphi(v) = sum(\varphi(u); \varphi(v))$



Embedding streams and par

Infinite streams and zip

Two A-streams $u : [0, +\infty[\to A$ and $v : [0, +\infty[\to B$ and their zip

$$u|v : [0, +\infty[\to A \times B$$

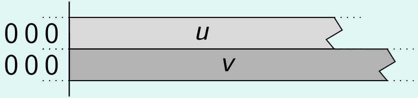
defined by

$$u|v(k) = (u(k), v(k))$$

for every $0 \leq k$.

$$\psi : \mathit{streams} \to \mathit{tiledStreams}$$

Streams embedding : $\psi(u)|\psi(v) = \psi(u); \psi(v)$



Embedding strings and streams with mixed product

Mixed product

An A-string $u : [0, d_u[\to A$ and an A-stream $v : [0, +\infty[\to A$ and their mixed product

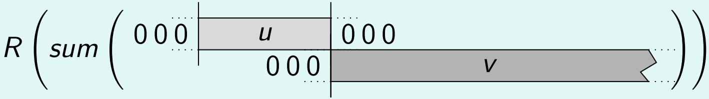
$$u :: v : [0, +\infty[\to A$$

defined by

$$u :: v(k) = \begin{cases} u(k) & \text{when } 0 \le k < d_u \\ v(k - d_u) & \text{when } d_u \le k \end{cases}$$

for every $0 \le k$.

Mixed embedding: $\varphi(u) :: \psi(v) = R(sum(\varphi(u); \psi(v)))$



5. T-calculus

A syntax for the tiled algebra and some type systems

Syntax

$p ::=$		*– primitive constructs –*
	$\mathtt{c}$	(constant)
$\mid$	$\mathtt{x}$	(variable)
$\mid$	$\mathtt{f}\,(p_1, p_2, \cdots, p_n)$	(function lifting)
$\mid$	$\mathtt{x} = p_1$	(declaration)
$\mid$	$\mathtt{R}\,(p_1)$	(sync. reset)
$\mid$	$\mathtt{L}\,(p_1)$	(sync. co-reset)
		– derived constructs –
$\mid$	$p_1 \text{ op } p_2$	(operator)
$\mid$	$p_1 \;;\; p_2$	(synchronization product)

Example

Assume A with operator $+$ with neutral element 0.

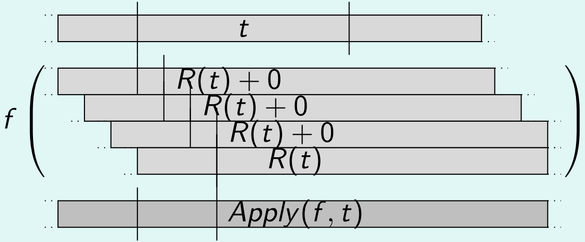
Sound processing

A sound processing function $f : A^n \to A$ on sliding windows of length $n \geq 1$. The constant A-tiled stream 0 with $d(0) = 1$.

$$Apply(f, t) = f(\underbrace{(R(t) + 0; R(t) + 0; \cdots ; R(t) + 0; R(t))}_{n \text{ times}})$$

for an arbitrary tiled stream t with $d(p) = n - 1$.

In picture with $n = 4$



Semantics (principle)

The semantic mapping

An environment $\mathcal{E}$ that maps variables to tiled streams.
An program interpretation $[\![p]\!]_{\mathcal{E}}$ in a tiled stream is sound when it satisfies the inductive rules (next slide) and the fixpoint rule:

(Y) *For every* $\mathtt{x}$ *occurring in* p *we have* $\mathcal{E}(\mathtt{x}) = [\![p_{\mathtt{x}}]\!]_{\mathcal{E}}$.

with $p_{\mathtt{x}}$ the unique definition of $\mathtt{x}$ in p.

Canonical fixpoint semantics (a little adhoc)

Start with silent interpretation (i.e. 0) for every variable and iterate semantics rules till a fixpoint is reach.

Semantics (inductive rules)

- Const.: $d(\llbracket \mathtt{c} \rrbracket_{\mathcal{E}}) = 0$ and $\llbracket \mathtt{c} \rrbracket_{\mathcal{E}}(k) = \begin{cases} \mathtt{c} & \text{when } k = 0, \\ 0 & \text{when } k \neq 0, \end{cases}$
- Variable: $\llbracket \mathtt{x} \rrbracket_{\mathcal{E}}(k) = \mathcal{E}(\mathtt{x})(k)$,
- Mapping: $d(\llbracket \mathtt{f}\,(p_1, \cdots, p_n) \rrbracket_{\mathcal{E}}) = \sum_{i \in [1,n]} d(\llbracket p_i \rrbracket_{\mathcal{E}})$
  and $\llbracket \mathtt{f}\,(p_1, \cdots, p_n) \rrbracket_{\mathcal{E}}(k) = \mathtt{f}(v_1, \cdots, v_n)$
  with $v_i = \llbracket p_i \rrbracket_{\mathcal{E}} \left(k - \sum_{1 \leq j < i} d(\llbracket p_j \rrbracket_{\mathcal{E}})\right)$,
- Decl.: $d(\llbracket \mathtt{x} = p_{\mathtt{x}} \rrbracket_{\mathcal{E}}) = d(\llbracket p_{\mathtt{x}} \rrbracket_{\mathcal{E}})$ and
  $\llbracket \mathtt{x} = p_{\mathtt{x}} \rrbracket_{\mathcal{E}}(k) = \llbracket p_{\mathtt{x}} \rrbracket_{\mathcal{E}}(k)$,
- Reset: $d(\llbracket \mathtt{R}\,(p_1) \rrbracket_{\mathcal{E}}) = 0$ and $\llbracket \mathtt{R}\,(p_1) \rrbracket_{\mathcal{E}}(k) = \llbracket p_1 \rrbracket_{\mathcal{E}}(k)$,
- Co-res.: $d(\llbracket \mathtt{L}\,(p_1) \rrbracket_{\mathcal{E}}) = 0$ and
  $\llbracket \mathtt{L}\,(p_1) \rrbracket_{\mathcal{E}}(k) = \llbracket p_1 \rrbracket_{\mathcal{E}}(k + d(\llbracket p_1 \rrbracket_{\mathcal{E}}))$

Typing I : basic types and sync. length inference

- Constants: $$\frac{}{\Gamma \vdash \mathtt{c} : (1, \alpha_c)}$$
- Variables: $$\frac{(\mathtt{x}, (d, \alpha)) \in \Gamma}{\Gamma \vdash \mathtt{x} : (d, \alpha)}$$
- Mapping: $$\frac{\Gamma \vdash p_i : (d_i, \alpha_i) \quad (i \in [1, n])}{\Gamma \vdash \mathtt{f}\,(p_1, \cdots, p_n) : (d_1 + \cdots + d_n, \alpha)}$$ with $\mathtt{f} : \alpha_1 \times \alpha_2 \times \cdots \times \alpha_n \to \alpha$
- Declaration: $$\frac{\Gamma \vdash \mathtt{x} : (d, \alpha) \quad \Gamma \vdash p : (d, \alpha)}{\Gamma \vdash \mathtt{x} = p : (d, \alpha)}$$
- Sync. reset: $$\frac{\Gamma \vdash p : (d, \alpha)}{\Gamma \vdash \mathtt{R}(p) : (0, \alpha)}$$
- Sync. co-reset: $$\frac{\Gamma \vdash p : (d, \alpha)}{\Gamma \vdash \mathtt{L}(p) : (0, \alpha)}$$

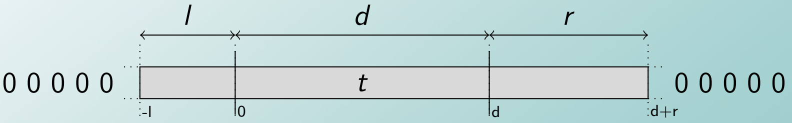
Typing II: Synchronization profile

Sync. profile definition

A tiled stream $t : \mathbb{Z} \to A$.
The triple $(l, d, r) \in \overline{\mathbb{N}} \times \mathbb{N} \times \overline{\mathbb{N}}$ is a *synchronization profile* for t when $d(t) = d$ and for every $k \in \mathbb{Z}$,

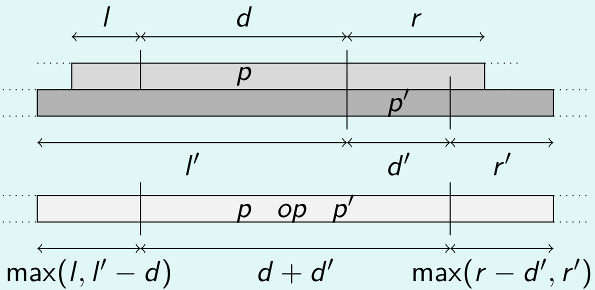
$$\text{if } t(f) \neq 0 \text{ then } -l \leq k \leq d + r$$

with $\overline{\mathbb{N}} = \mathbb{N} + \infty$ and $x + \infty = \infty + x = \infty$ for every $x \in \overline{\mathbb{N}}$.



Typing II: Induced monoid

Remark: *op*-product of two tiled streams



Lemma

The set $\overline{\mathbb{N}} \times \mathbb{N} \times \overline{\mathbb{N}}$ *with product*

$$(l, d, r) \cdot (l', d', r') = (\max(l, l' - d), d + d', \max(r - d', r'))$$

is related with the free inverse monoid with one generator [5, 6] with neutral element $(0, 0, 0)$.

Typing II: Synchronization profile rules

- Constants:
$$\frac{}{\Delta \vdash \mathtt{c} : (0, 1, 0)}$$
- Variables:
$$\frac{(\mathtt{x}, (l, d, r)) \in \Delta}{\Delta \vdash \mathtt{x} : (l, d, r)}$$
- Mapping:
$$\frac{\Delta \vdash p_i : (l_i, d_i, r_i) \quad (i \in [1, n])}{\Delta \vdash \mathtt{f}\,(p_1, \cdots, p_n) : (l, d, r)}$$

with $l = \max\left(l_i - \sum_{1 \leq j < i} d_j\right)$, $d = \sum_i d_i$,

and $r = \max\left(r_i - \sum_{i < j \leq n} d_j\right)$,

- Declaration:
$$\frac{\Delta \vdash \mathtt{x} : (l, d, r) \quad \Delta \vdash p : (l, d, r)}{\Delta \vdash \mathtt{x} = p : (l, d, r)}$$
- Sync. reset:
$$\frac{\Delta \vdash p : (l, d, r)}{\Delta \vdash \mathtt{R}(p) : (l, 0, d + r)}$$
- Sync. co-reset:
$$\frac{\Delta \vdash p : (l, d, r)}{\Delta \vdash \mathtt{L}(p) : (l + d, 0, r)}$$

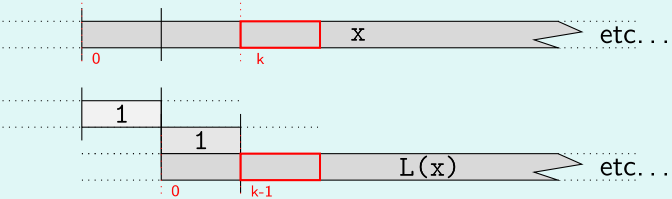
6. Transducers

When computing is easy

Operational semantics: example

Example

A program p defined by $\mathtt{x} = \underbrace{\mathtt{1} + \mathtt{R}(\mathtt{1} + \mathtt{L}(\mathtt{x}))}_{p_{\mathtt{x}}}$



with x indices marked in red. And, for every $k \in \mathbb{Z}$, we have: $[\![p_{\mathtt{x}}]\!]_{\mathcal{E}}(k) = 1 + [\![x]\!]_{\mathcal{E}}(k + -1)$ and $\delta(p_{\mathtt{x}}, \mathtt{x}) = \{-1\}$.

Remark

Iterative semantics is thus uniquely determined by (1) 0 *everywhere in the past* and (2) a *computation rule compatible with time flow*.

Operational semantics: temporal dependencies

Definition

We look for

$$\delta : \mathit{Program} \times \mathit{Variables} \to \mathit{Offsets}$$

such that, for every program

$$p = t(x_1, x_2, \cdots, x_n)$$

there exists a function

$$f_p : \prod\{\alpha_{x_i} : 1 \leq i \leq n, d \in \delta(p, x_i)\} \to \alpha_p$$

such that, for every “good” valuation $\mathcal{E}$:

$$[\![p]\!]_{\mathcal{E}}(k) = f_p\left(\{[\![x_i]\!]_{\mathcal{E}}(k + d) : 1 \leq i \leq n, d \in \delta(p, x_i)\}\right)$$

for every $k \in \mathbb{Z}$.

Operational semantics: direct temporal dependencies

Direct temporal dependencies rules

- Constants: $\delta(\mathtt{c}, \mathtt{x}) = \emptyset$,
- Variable: $\delta(\mathtt{y}, \mathtt{x}) = \emptyset$ when $\mathtt{x}$ and $\mathtt{y}$ are distinct and $\delta(\mathtt{x}, \mathtt{x}) = \{0\}$ otherwise,
- Mapping: $\delta(\mathtt{f}\,(p_1, \cdots, p_n), \mathtt{x}) =$

$$\bigcup_{1 \leq i \leq n} \left(\delta(p_i, \mathtt{x}) - \sum_{1 \leq j < i} d(p_j) \right),$$

- Declaration: $\delta(\mathtt{y} = p_{\mathtt{y}}, \mathtt{x}) = \delta(\mathtt{y}, \mathtt{x})$,
- Sync. reset: $\delta(\mathtt{R}\,(p_1), \mathtt{x}) = \delta(p_1, \mathtt{x})$,
- Sync. co-reset: $\delta(\mathtt{L}\,(p_1), \mathtt{x}) = \delta(p_1, \mathtt{x}) + d(p_1)$.

Operational semantics: iterated temporal dependencies

Definition

For every program p, every variable x that occurs in p, every subprogram q of p, let:

$$\boxed{\delta^*(q,\mathrm{x}) = \delta(q,\mathrm{x}) \cup \bigcup_{\mathrm{y}\in\mathcal{X}_p} \left(\delta(q,\mathrm{y}) + \delta^*(p_{\mathrm{y}},\mathrm{x})\right)}$$

with $X + Y = \{x + y \in \mathbb{Z} : x \in X, y \in Y\}$.

Theorem

Assume that for every variable x that occurs in p the set $\delta^*(\mathrm{x},\mathrm{x})$ only contained strictly negative values.
Then the program p admits an iterative semantics that is causal and with finite past.
Moreover, it can be compiled into a finite state synchronous transducer/mealy machine [7].

7. Conclusion

Summary

- Programming time with tiled streams,
- The underlying algebra (SEQ, RESET, CO-RESET),
- A type system for causality and effective start,
- A finite state operational semantics (mealy machine).

Dynamical tilings

Question

Computing sync. length out of RT input via input monitoring ?
Induced conditionals and loops ?
Multi-tempi semantics ?

- Distinguishing active tiled streams, e.g. score follower,
- Passive/reactive tiled streams, e.g. generated tiled streams.

[1] D. Janin, "Vers une modélisation combinatoire des structures rythmiques simples de la musique," *Revue Francophone d'Informatique Musicale (RFIM)*, vol. 2, 2012.

[2] F. Berthaut, D. Janin, and B. Martin, "Advanced synchronization of audio or symbolic musical patterns: an algebraic approach," *International Journal of Semantic Computing*, vol. 6, no. 4, pp. 409–427, 2012.

[3] P. Hudak, "A sound and complete axiomatization of polymorphic temporal media," Tech. Rep. RR-1259, Department of Computer Science, Yale University, 2008.

[4] D. Perrin and J.-E. Pin, *Infinite Words: Automata, Semigroups, Logic and Games*, vol. 141 of *Pure and Applied Mathematics*. Elsevier, 2004.

[5] M. V. Lawson, "McAlister semigroups," *Journal of Algebra*, vol. 202, no. 1, pp. 276 – 294, 1998.

[6] M. V. Lawson, *Inverse Semigroups : The theory of partial symmetries.*
World Scientific, 1998.

[7] J. Sakarovitch, *Elements of Automata Theory.*
Cambridge University Press, 2009.