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:

\[\text{(a)}\]
String modeling (b)
Problem:

- we have inserted rests of various size: 5, 5, 1 and 8,
- we have lost the logical structure \(3 \times (a) + (b)\),
- 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 !
Embedding (audio) strings and (audio) streams into tiled streams [2]
Tiled streams

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

Tiled stream

A \textit{“bi-infinite” sequence} of values $t : \mathbb{Z} \to A$ with an additional \textit{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 $t : \mathbb{Z} \rightarrow A$ with an additional synchronization length $d(t) \in \mathbb{N}$. 
Tiled stream product: the “free” case

Tiled stream product

Two tiled stream \( t_1 : \mathbb{Z} \rightarrow A \) and \( t_2 : \mathbb{Z} \rightarrow B \) and their product \( t_1; t_2 : \mathbb{Z} \rightarrow A \times B \) 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 \( op : A \times B \rightarrow C \)

**Operator lifting**

Two tiled streams \( t_1 : \mathbb{Z} \rightarrow A \) and \( t_2 : \mathbb{Z} \rightarrow B \) let \( t_1 \circ t_2 : \mathbb{Z} \rightarrow C \)
defined by \( d(t_1 \circ t_2) = d(t_1) + d(t_2) \) and

\[
(t_1 \circ t_2)(k) = t_1(k) \circ t_2(k - d(t_1))
\]

for every \( k \in \mathbb{Z} \).

**Synchronization + Fusion : \( t_1 \circ t_2 \)**
Tiled stream product: the parameterized case

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

**Operator lifting**

Two tiled streams $t_1 : \mathbb{Z} \rightarrow A$ and $t_2 : \mathbb{Z} \rightarrow B$ let $t_1 \ op \ t_2 : \mathbb{Z} \rightarrow C$ 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 $t : \mathbb{Z} \rightarrow A$, 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

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

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:

\[ \text{fork}(t_1, t_2) = R(t_1); t_2 \]

\[ \text{join}(t_1, t_2) = t_1; L(t_2) \]

with synchronization length \( d(\text{fork}(t_1, t_2)) = d(t_2) \) and \( d(\text{join}(t_1, t_2)) = d(t_1) \).
4. Embeddings

Links with Hudak’s *Polymorphic Temporal Media* [3] and $\omega$-semigroups [4]
Embedding strings and concatenation

Finite strings and concatenation

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

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

defined by

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

\(\varphi : strings \rightarrow tiledStreams\)

String embedding: \(\varphi(u) \cdot \varphi(v) = \text{sum}(\varphi(u); \varphi(v))\)
Infinite streams and zip

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

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

defined by

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

for every $0 \leq k$.

$$\psi : \text{streams} \rightarrow \text{tiledStreams}$$

Streams embedding : $\psi(u) \| \psi(v) = \psi(u); \psi(v)$
Embedding strings and streams with mixed product

**Mixed product**

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

$$u :: v : [0, +\infty] \rightarrow A$$

defined by

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

for every $0 \leq k$.

**Mixed embedding:** $\varphi(u) :: \psi(v) = R(\text{sum}(\varphi(u); \psi(v)))$

$$R \left( \text{sum} \left( \begin{array}{c} 0 \ 0 \ 0 \\ u \\ 0 \ 0 \ 0 \end{array} \right) \right)$$
A syntax for the tiled algebra and some type systems
Syntax

\[ p ::= \]

- **primitive constructs** -
  \[ c \]
  (constant)
  \[ x \]
  (variable)
  \[ f(p_1, p_2, \ldots, p_n) \]
  (function lifting)
  \[ x = p_1 \]
  (declaration)
  \[ R(p_1) \]
  (sync. reset)
  \[ L(p_1) \]
  (sync. co-reset)

- **derived constructs** -
  \[ p_1 \text{ op } p_2 \]
  (operator)
  \[ p_1 ; p_2 \]
  (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 $d(0) = 1$.

$$Apply(f, t) = f((R(t) + 0; R(t) + 0; \cdots ; R(t) + 0; R(t)))$$

$n$ times

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

**In picture with $n = 4$**

\[ f \left( \begin{array}{c}
R(t) + 0 \\
R(t) + 0 \\
R(t) + 0 \\
R(t)
\end{array} \right) \]

\[ Apply(f, t) \]
The semantic mapping

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

\[
\text{(Y) For every } x \text{ occurring in } p \text{ we have } \mathcal{E}(x) = \llbracket p_x \rrbracket_\mathcal{E}.
\]

with $p_x$ the unique definition of $x$ 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.**: $d([c]_E) = 0$ and $[c]_E(k) = \begin{cases} c & \text{when } k = 0, \\ 0 & \text{when } k \neq 0, \end{cases}$

- **Variable**: $[x]_E(k) = E(x)(k)$,

- **Mapping**: $d([f(p_1, \ldots, p_n)]_E) = \sum_{i \in [1, n]} d([p_i]_E)$
  
  and $[f(p_1, \ldots, p_n)]_E(k) = f(v_1, \ldots, v_n)$
  
  with $v_i = [p_i]_E \left( k - \sum_{1 \leq j < i} d([p_j]_E) \right)$,

- **Decl.**: $d([x = p_x]_E) = d([p_x]_E)$ and
  
  $[x = p_x]_E(k) = [p_x]_E(k)$,

- **Reset**: $d([R(p_1)]_E) = 0$ and $[R(p_1)]_E(k) = [p_1]_E(k)$,

- **Co-res.**: $d([L(p_1)]_E) = 0$ and
  
  $[L(p_1)]_E(k) = [p_1]_E(k + d([p_1]_E))$
Typing I: basic types and sync. length inference

- **Constants:** \( \Gamma \vdash c : (1, \alpha_c) \)

- **Variables:** \( (x, (d, \alpha)) \in \Gamma \) \( \Gamma \vdash x : (d, \alpha) \)

- **Mapping:** \( \Gamma \vdash p_i : (d_i, \alpha_i) \) \((i \in [1, n])\) \( \Gamma \vdash f(p_1, \ldots, p_n) : (d_1 + \cdots + d_n, \alpha) \)
  with \( f : \alpha_1 \times \alpha_2 \times \cdots \times \alpha_n \to \alpha \)

- **Declaration:** \( \Gamma \vdash x : (d, \alpha) \) \( \Gamma \vdash p : (d, \alpha) \) \( \Gamma \vdash x = p : (d, \alpha) \)

- **Sync. reset:** \( \Gamma \vdash p : (d, \alpha) \) \( \Gamma \vdash R(p) : (0, \alpha) \)

- **Sync. co-reset:** \( \Gamma \vdash p : (d, \alpha) \) \( \Gamma \vdash L(p) : (0, \alpha) \)
Sync. profile definition

A tiled stream \( t : \mathbb{Z} \rightarrow A \).

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

\[
\begin{align*}
(l, d, r) \cdot (l', d', r') &= (\max(l, l' - d), d + d', \max(r - d', r'))
\end{align*}
\]

is related with the free inverse monoid with one generator [5, 6] with neutral element (0, 0, 0).
Typing II: Synchronization profile rules

- **Constants:**
  \[ \Delta \vdash c : (0, 1, 0) \]
  \[ (x, (l, d, r)) \in \Delta \]

- **Variables:**
  \[ \Delta \vdash x : (l, d, r) \]

- **Mapping:**
  \[ \Delta \vdash p_i : (l_i, d_i, r_i) \quad (i \in [1, n]) \]
  \[ \Delta \vdash f(p_1, \ldots, p_n) : (l, d, r) \]
  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:**
  \[ \Delta \vdash x = p : (l, d, r) \]
  \[ \Delta \vdash x : (l, d, r) \]
  \[ \Delta \vdash p : (l, d, r) \]

- **Sync. reset:**
  \[ \Delta \vdash p : (l, d, r) \]
  \[ \Delta \vdash R(p) : (l, 0, d + r) \]

- **Sync. co-reset:**
  \[ \Delta \vdash p : (l, d, r) \]
  \[ \Delta \vdash L(p) : (l + d, 0, r) \]
6. Transducers

When computing is easy
Operational semantics: example

Example

A program $p$ defined by $x = 1 + R(1 + L(x))$

\[ p_x \]

with $x$ indices marked in red. And, for every $k \in \mathbb{Z}$, we have:

\[ \llbracket p_x \rrbracket_\varepsilon(k) = 1 + \llbracket x \rrbracket_\varepsilon(k + -1) \] and $\delta(p_x, x) = \{-1\}$.

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

\[ \delta : \text{Program} \times \text{Variables} \rightarrow \text{Offsets} \]

such that, for every program

\[ p = t(x_1, x_2, \cdots, x_n) \]

there exists a function

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

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

\[ \llbracket p \rrbracket \mathcal{E}(k) = f_p (\{ \llbracket x_i \rrbracket \mathcal{E}(k + d) : 1 \leq i \leq n, d \in \delta(p, x_i) \}) \]

for every \( k \in \mathbb{Z} \).
Operational semantics: direct temporal dependencies

Direct temporal dependencies rules

- Constants: \( \delta(c, x) = \emptyset \),
- Variable: \( \delta(y, x) = \emptyset \) when \( x \) and \( y \) are distinct and \( \delta(x, x) = \{0\} \) otherwise,
- Mapping: \( \delta(f(p_1, \ldots, p_n), x) = \)
  \[
  \bigcup_{1 \leq i \leq n} \left( \delta(p_i, x) - \sum_{1 \leq j < i} d(p_j) \right),
  \]
- Declaration: \( \delta(y = p_y, x) = \delta(y, x) \),
- Sync. reset: \( \delta(R(p_1), x) = \delta(p_1, x) \),
- Sync. co-reset: \( \delta(L(p_1), x) = \delta(p_1, x) + d(p_1) \).
Operational semantics: iterated temporal dependencies

Definition

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

$$\delta^*(q, x) = \delta(q, x) \cup \bigcup_{y \in X_p} (\delta(q, y) + \delta^*(p_y, x))$$

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 $\delta^*(x, x)$ 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.


