Tiled Polymorphic Temporal Media

Paul Hudak
Dept. of Computer Science
Yale University

David Janin
LaBRI
University of Bordeaux

FARM’14
Gothenburg, Sweden
September 6, 2014
Motivation

Our goal is to design a system for combining multimedia objects (such as sound, video, and animation) that is:

- **Simple** (a beginner can use it)
- **Efficient** (runs fast, efficient transformations)
- **Elegant** (concise, perspicuous)
- **Sound** (strong mathematical properties)
One Approach: Polymorphic Temporal Media

- **Polymorphic Temporal Media** (PTM) [Hudak ‘04,’08] has four key elements:
  - A *neutral* value (e.g. silence or transparency)
  - A set of *primitive* values (e.g. notes or video frames)
  - A *binary sequential composition operator* `:+:` such that `p1 :+: p2` is “p1 followed by p2.”
  - A *binary parallel composition operator* `:=:` such that `p1 :=: p2` is “p1 in parallel with p2.”
- PTM is the basis of Haskore [Hudak ‘96] and Euterpea [Hudak ‘13]
PTM Properties

- Nice algebraic properties:
  - 
  - \((p1 :+: p2) :+: p3 == p1 :+: (p2 :+: p3)\)
  - \((p1 :=: p2) :=: p3 == p1 :=: (p2 :=: p3)\)
  - \(p1 :=: p2 == p2 :=: p1\)
  - \(\text{rest } 0 :+: p == p == p :+: \text{rest } 0\)
  - if \(\text{dur } p1 = \text{dur } p3\) then
    - \((p1 :+: p2) :=: (p3 :+: p4) ==\)
    - \((p1 :=: p3) :+: (p2 :=: p4)\)

- Indeed, there exists a set of axioms that is *sound* and *complete* [Hudak ’08].

- PTM is *polymorphic* (sound, music, video, animation, robot movement).

- PTM is also *efficient*, and arguably *simple* and *elegant*. 
PTM Shortcomings

But all is not rosy... PTM has certain shortcomings:

- **Semantics:**
  p1 :=: p2 has several interpretations: p1 and p2 may start at the same time, or end at the same time, or be centered in time. None are inherently the right choice.

- **Expressiveness:**
  PTM doe not distinguish between “logical” and “actual” start and end times. For example:
  - In music, a measure may have a “pickup” (or *anacrusis*) – logically, it begins when the measure begins, but actually it begins when the pickup begins.
  - In video, a clip may have a “fade-in” – logically the beginning is the main clip, but actually it is the fade-in.
A Solution: *Tiled PTM*

- At FARM ‘13 David Janin presented the *T-Calculus* which, at least conceptually, eliminates the shortcomings of PTM.
- It can be viewed as *tiling* the one dimension of time.
- However, it has its own shortcomings:
  - It has no implementation.
  - It is constrained to the power of finite state transducers.
  - Its connection to inverse semi-group theory is interesting, but weaker than it could be.
  - Its recursion scheme (for infinite terms) is weak.
- Our solution:

  \[ \text{T-Calculus} + \text{PTM} = \text{Tiled PTM} \]
Contributions

- The design of *Tiled PTM* (T-PTM)
  - Combines best attributes of PTM and the T-Calculus
- Discovery of new *algebraic properties*
  - In particular, a stronger connection to inverse semi-groups
- Exploration of effective *recursion schemes*
  - Allows infinite tilings
- An *implementation* in Haskell
  - Specifically, in Euterpea (i.e. PTM constrained to music)
Basic Idea

- A tiled PTM has two *synchronization marks*:
  - *pre* marks the *logical start*, relative to the actual start
  - *post* marks the *logical end*, also relative to the actual start

- Pictorially:
Tiled Product

- Two tiled PTM values can be combined by a binary *tiled product* operator \( \% \).
- \( m_1 \% m_2 \) is a tiled PTM that is the tiled product of \( m_1 \) and \( m_2 \).
- This involves:
  - *Synchronization* of the logical start of \( m_2 \) with the logical end of \( m_1 \).
  - *Fusion* of the overlapping content of \( m_1 \) and \( m_2 \).
- Pictorially: [next slide]
Some Key Points

- In the construction of a tiled product, *partial overlap* may occur.
- So it is neither a sequential product nor a parallel product – *it is both*.
- `;++` and `==:` can be encoded in terms of `%`.

- In a given tile, *pre may be greater than post*!
  - In which case, what is the meaning of tiled product?
Algebraic Properties

- With a suitable notion of *observational equivalence* (see paper) we can show:

  \[(t_1 \% t_2) \% t_3 \equiv t_1 \% (t_2 \% t_3)\]

- T-PTM’s *neutral* (or *silent*) tile \(r^d\) has duration \(d\), and:

  \[r^0 \% t \equiv t \equiv t \% r^0\]

- Therefore T-PTM is a *monoid*. 
Other Operators

- **Primitive monomorphic values:**
  - In music, \( t n o d \) is a musical note with pitch class \( n \), octave \( o \), and duration \( d \). E.g. \( t C 5 \ (1/4) \) is middle C with quarter note duration.

- Reset \( re \)
- Co-reset \( co \)
- Inverse \( inv \)

[ let’s look at pictorial descriptions ]
Example of Modularity

- let \( m_1 = pm :+: r\ d_0 :+:\ pu \) in \( m_1 :+: m_2 \)

- Now suppose we lengthen \( pu \):
  - let \( m_1 = pm :+: r\ d_1 :+:\ pu \) in \( m_1 :+: m_2 \)

- Suppose \( pu \) becomes sufficiently large:
  - let \( m_{1a} = pm \)
  - \( m_{1b} = r\ d_3 :+:\ pu \)
  - in \( (m_{1a} :=: m_{1b}) :+: m_2 \)

- More modular, but still lacks logical structure

- In contrast, with T-PTM:
  - \( m_1 % (co\ pu % m_2) \)

- Fully modular: \textit{changes to pu induce no changes to m1 or m2.}

- Has logical structure
Why inv? Why “negative” tiles?

- Reset and co-reset can be defined in terms of inv:
  - \( \text{ret } t = t \% \text{inv } t \)
  - \( \text{cot } t = \text{inv } t \% t \)

- In an inverse semi-group the inverse of an element \( x \) is an element \( y \) such that:
  - \( x \cdot y \cdot x = x \) and \( y \cdot x \cdot y = y \)

Now note that in T-PTM:
- \( t \% \text{inv } t \% t = t \) and \( \text{inv } t \% t \% \text{inv } t = \text{inv } t \)

- Therefore, \( T\text{-PTM is an inverse semi-group.} \)

- Using inverse semi-group theory, various properties of T-PTM are immediate (see paper).
Recursive and Infinite PTM

- To *render* a temporal value, one needs to incrementally enumerate its instantaneous values over time.
- With PTM, this is straightforward, even for recursively defined, infinite values:
  \[ m = \text{c 4 en} :+: m \]
- Even parallel composition is OK:
  \[ m1 :=: m2 \]
  - Render m1
  - Render m2
  - Time-merge the results
Recursive and Infinite Tiles

- But there is a problem with:
  \[ x = t \cdot c \cdot 4 \cdot \text{en} \cdot \% \cdot x \]
  because *the value of post is infinite.*

- Even this version is problematical:
  \[ x = t \cdot c \cdot 4 \cdot \text{en} \cdot \% \cdot \text{re} \cdot x \]

- In general we cannot always render \( t1 \% t2 \) because *the anacrusis of t2 may begin before pre t1.*

- By defining a new operator \( \%\backslash \) that *ignores* such an anacrusis, we make progress in that:
  \[ x = t \cdot c \cdot 4 \cdot \text{en} \cdot \%\backslash \cdot \text{re} \cdot x \]
  can be rendered properly.

- But note: \( \%\backslash \) is *not associative.*
In the paper...

- Further exploration of recursive tiles of form $t = f t$.
  - Relies on special fixpoint operator
- Implementation of T-PTM in Euterpea (PTM)
  - Serves as specification
- Observational equivalence.
- Formal connection between $(:+:,:=:)$ and $\%$.
- Other operators: `resynch`, `coresynch`, `stretch`, `costretch`.
- Examples.
Thank You!

Any questions?