

Towards a Higher-Dimensional String Theory for the Modeling of Computerized Systems

David Janin
LaBRI, Bordeaux INP,
University of Bordeaux

SOFSEM 2014
40th International Conference on
Current Trends in Theory and Practice of Computer Science
January 25-30,
Nový Smokovec, High Tatras,
Slovakia

Plan of the talk

- I. Modeling music: from opera to bebop.
- II. From birooted words to higher dimensional strings.
- III. A bit of dinning philosophy.
- IV. Towards a language theory of higher dimensional strings.
 - a. Definable languages.
 - b. Walking automata.
 - c. Non deterministic automata.
- V. Quasi-recognizable languages.
- VI. Conclusion: back to music.

1. Music modeling

From opera, to arabesques down to bebop.

Opera is the only medium in which everyone can be talking at the same time and still manage to understand each other. . .

Amadeus, XVIIIth century,

(from M. Forman in eponym movie)

Centuries later, this has become common practice in communication networks.

Question

Are there other musical *techniques* or *metaphors* that can be used in computer systems modeling ?

A musical example

The image displays two systems of musical notation for a piano piece. The first system is marked "a Tempo" and "p" (piano). It features a treble and bass clef with a key signature of two sharps (F# and C#). The music consists of flowing, arpeggiated patterns with triplets and slurs. The second system continues the piece, showing a change in time signature to 4/4 in the final measure. The notation includes various rhythmic values, slurs, and dynamic markings.

Extracted from Debussy's arabesques : *andantino con moto*

The image displays a musical score for a soprano voice and piano accompaniment. The score is written in treble clef with a key signature of three sharps (F#, C#, G#) and a time signature of 4/4. The tempo is marked "a Tempo" and the dynamics are marked "p". The score consists of two systems. The first system shows the soprano voice line and the piano accompaniment. The second system continues the music. Red boxes highlight specific motifs in the soprano voice line, illustrating interleaved motifs. The first red box highlights a motif starting at the beginning of the second system. The second red box highlights a motif starting at the beginning of the third system. The third red box highlights a motif starting at the beginning of the fourth system. The piano accompaniment features complex rhythmic patterns, including triplets and sixteenth notes.

interleaved motifs in soprano voice,




The image displays a musical score with two systems. The first system is marked "a Tempo" and "p". The score consists of a treble clef staff and a bass clef staff. The treble staff contains a melodic line with triplets and a long slur. The bass staff contains a rhythmic accompaniment with eighth notes and slurs. Annotations include a red box around the first measure of the treble staff, a red box around the second measure of the treble staff, and a red box around the third measure of the treble staff. A yellow box highlights the melodic line in the treble staff across the first two measures of the second system. A yellow box also highlights the melodic line in the treble staff across the first two measures of the first system.

an **underlying frame** in the medium,

The image displays a musical score with two systems. The first system consists of a treble clef staff and a bass clef staff, both in a key signature of three sharps (F#, C#, G#). The tempo is marked "a Tempo" and the dynamics are marked "p". The score features several triplet markings (indicated by a '3' over a group of notes). The second system continues the piece. Annotations include red boxes highlighting specific melodic lines in the treble clef staff and green boxes highlighting arpeggiated chords in the bass clef staff. The arpeggios are shown as individual notes of a chord being played in sequence, which is a technique used to make the harmonic structure more audible.

arpeggios to hear the harmony,



The image displays a musical score with two systems of staves. The top system consists of a treble clef staff and a bass clef staff. The bottom system also consists of a treble clef staff and a bass clef staff. The score is annotated with colored boxes: red boxes highlight specific melodic phrases in the treble staves, and green boxes highlight corresponding phrases in the bass staves. A blue line is drawn across the bottom system, indicating a bass line. The score includes dynamic markings such as *p* and *a Tempo*, and features triplets and slurs.

and a **bass line**...

What modeling language for music ?

A language for musical analysis (learning)

partition \implies *structure* \implies *intention*

A language for musical writing (or musical programming)

partition \longleftarrow *structure* \longleftarrow *intention*

An expressive language

With **sequential** and **parallel** composition operators
with **partial overlaps**, and **hierarchical description mechanisms**...

A usable language

with basic good programming properties such as **compositionality**,
reusability, etc. ...

My little blue suede shoes (Ch. Parker)



The image displays two staves of musical notation in G-flat major (two flats) and common time (C). The first staff contains three measures of music. The first two measures are grouped by a red bracket labeled '(a)', representing the first two exposures of a motif. The third measure is a rest. The second staff contains four measures. The first two measures are grouped by a red bracket labeled '(a)', representing the first two exposures of the motif. The third measure is a rest, and the fourth measure is a conclusive variation labeled '(b)', which features a different rhythmic pattern and a final note.

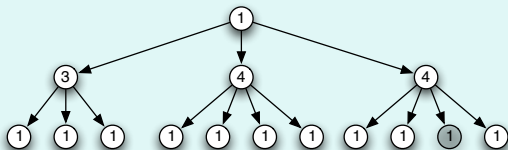
Analysis

Three exposures of the same motif (a) followed by a conclusive variation (b)

Modeling motiv (a)



modeled by

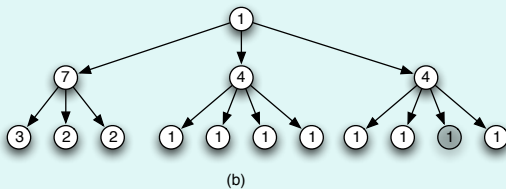


(a)

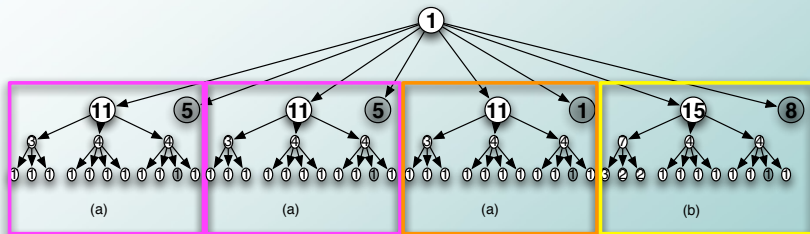
Modeling motiv (b)



modeled by



Resulting modeling



Drawbacks:

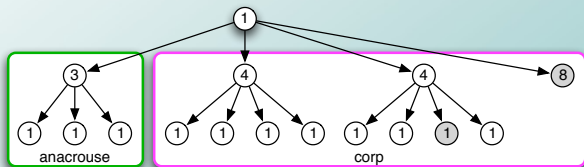
- insertion of silences with various length,
- logical structure $(3 \times (a) + (b))$ rather lost.

Alternative approach

Explicit first beat anticipation (anacrusis) [Jan13b]:



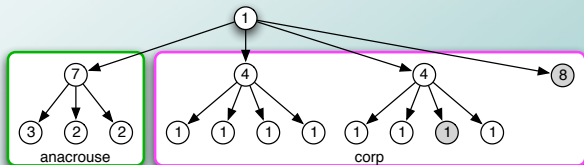
and fill with silence till the logical end (second bars):



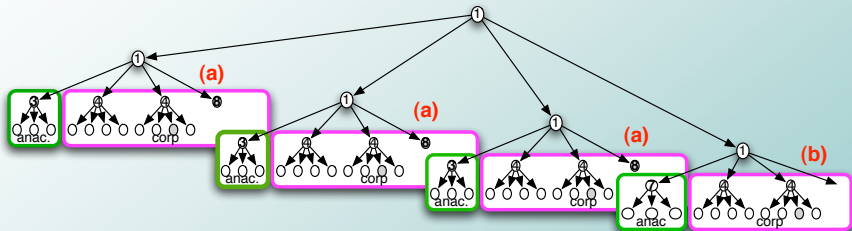
do the same for the second motif:



which gives:



The resulting *sequential composition*:



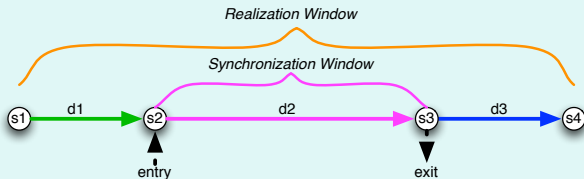
with *local overlaps*.

Here comes back the logical structure: $3 \times (a) + (b)$!

Synchronization vs realization windows

Idea [Jan12b]

Distinguish for every musical pattern:



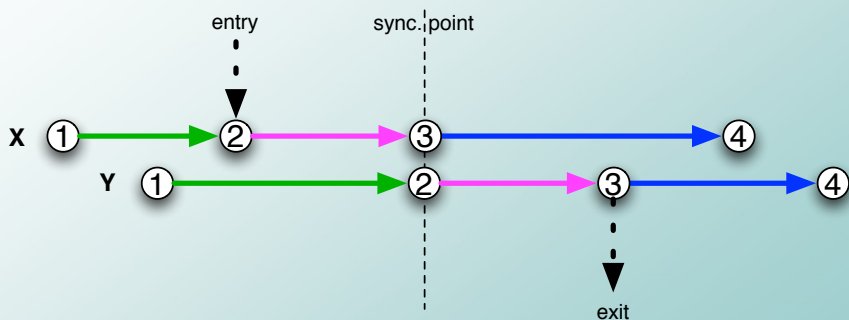
Old idea

Already implicitly present in musical modeling with **LOCO** [DH88], but somehow basic in **automation** or **software systems**.

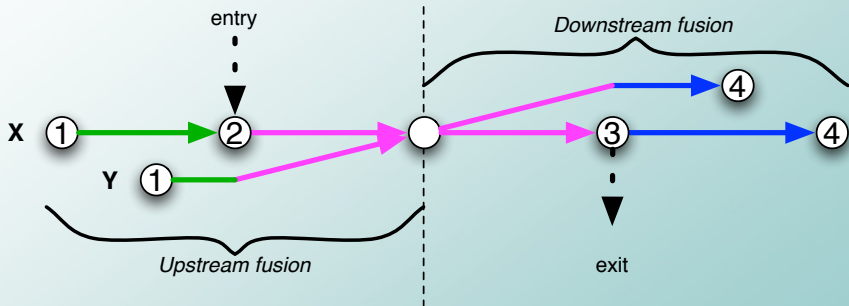
Remark

In theory of music: such a distinction generalizes bars and metrics. . .

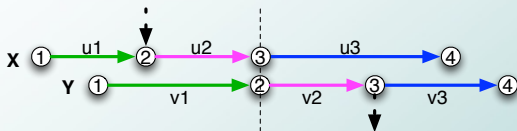
Pattern product : 1. synchronisation



Pattern product : 2. fusion



Induced algebra : the continuous case



Synchronization structures : $D_A = \mathbb{R} \times \mathbb{R} \times \mathbb{R}$, with product

$$\underbrace{(x_1, x_2, x_3)}_X \cdot \underbrace{(y_1, y_2, y_3)}_Y = \underbrace{(\max(x_1, y_1 - x_2), x_2 + y_2, \max(y_3, x_3 - y_2))}_{X.Y}$$

with fusion of underlying audio or musical patterns defined by **mixing and crossfading**. In practice, we obtain an music pattern algebra [BJM12], later turned into a prototype software [JBD13].

Theorem

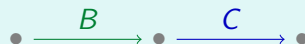
We obtain a monoid, i.e. a set equipped with an **associative product** and a **neutral element**.

2. Formal Models

Models for (abstract description of) music

Starting from the known world : a word based model

Musical objects



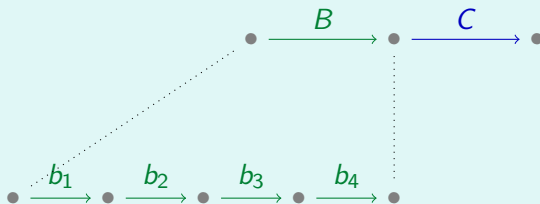
fairly commons in musical refinement

Even better : object to language refinement

Refine abstract objects by languages of possible realizations :
 $B \rightsquigarrow L_B$ and $C \rightsquigarrow L_C$ with **compatibility** constrains to filter
unwanted combinations.

Starting from the known world : a word based model

Musical objects refinement



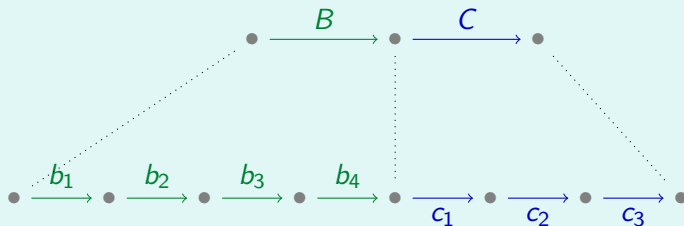
fairly commons in musical refinement

Even better : object to language refinement

Refine abstract objects by languages of possible realizations :
 $B \rightsquigarrow L_B$ and $C \rightsquigarrow L_C$ with **compatibility** constrains to filter unwanted combinations.

Starting from the known world : a word based model

Musical objects refinement



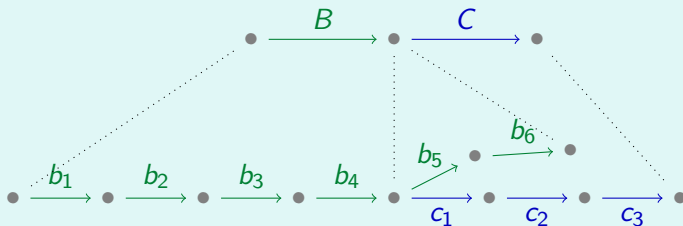
fairly commons in musical refinement

Even better : object to language refinement

Refine abstract objects by languages of possible realizations :
 $B \rightsquigarrow L_B$ and $C \rightsquigarrow L_C$ with **compatibility** constrains to filter
 unwanted combinations.

Starting from the known world : a word based model

Musical objects refinement with overlaps



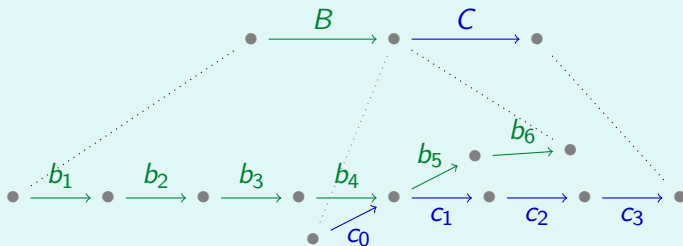
fairly commons in musical refinement

Even better : object to language refinement

Refine abstract objects by languages of possible realizations :
 $B \rightsquigarrow L_B$ and $C \rightsquigarrow L_C$ with compatibility constrains to filter unwanted combinations.

Starting from the known world : a word based model

Musical objects refinement with overlaps



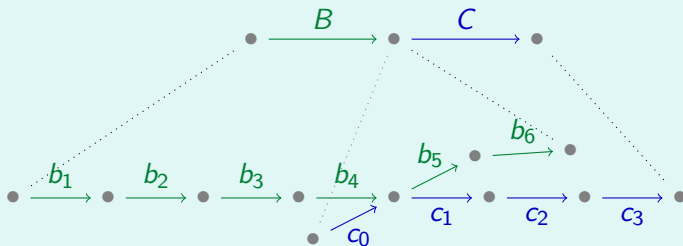
fairly commons in musical refinement

Even better : object to language refinement

Refine abstract objects by languages of possible realizations :
 $B \rightsquigarrow L_B$ and $C \rightsquigarrow L_C$ with compatibility constrains to filter
 unwanted combinations.

Starting from the known world : a word based model

Musical objects refinement with overlaps



fairly commons in musical refinement

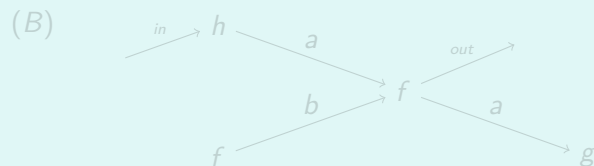
Even better : object to language refinement

Refine abstract objects by languages of possible realizations :
 $B \rightsquigarrow L_B$ and $C \rightsquigarrow L_C$ with **compatibility constrains** to filter unwanted combinations.

Labeled Munn's trees (or birooted F -trees)

Birooted F -tree

Let A and F two finite alphabets. F -labeled vertices and A -labeled edges

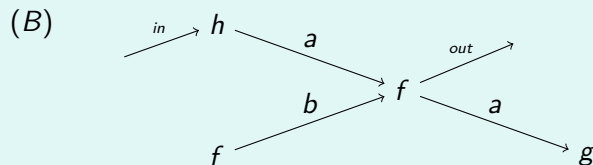


with input root (in) and output root (out).

Labeled Munn's trees (or birooted F -trees)

Birooted F -tree

Let A and F two finite alphabets. F -labeled vertices and A -labeled edges

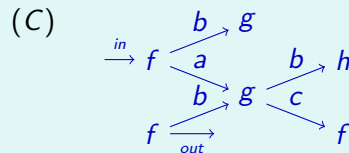
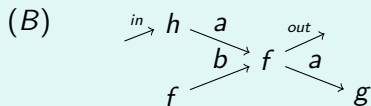


with input root (in) and output root (out).

Birooted F -trees product

A rich algebraic structure:

Product



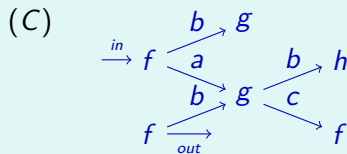
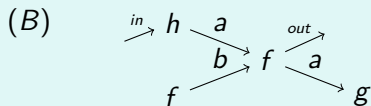
Theorem

Extended with 0 for the undefined case and 1 if needed, the resulting algebra is a *monoid* $\mathcal{B}(F)$.

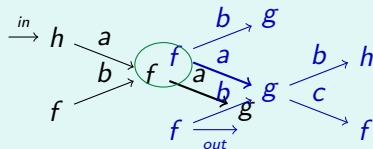
Birooted F -trees product

A rich algebraic structure:

Product = sync



SYNC



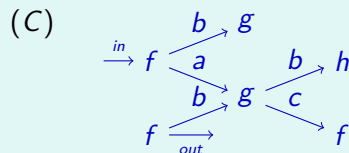
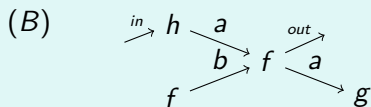
Theorem

Extended with 0 for the undefined case and 1 if needed, the resulting algebra is a monoid $\mathcal{B}(F)$.

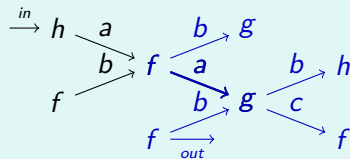
Birooted F -trees product

A rich algebraic structure:

Product = sync + fusion



FUSION



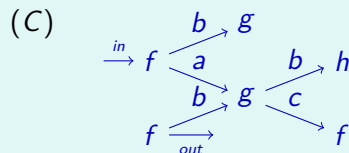
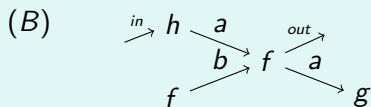
Theorem

Extended with 0 for the undefined case and 1 if needed, the resulting algebra is a monoid $\mathcal{B}(F)$.

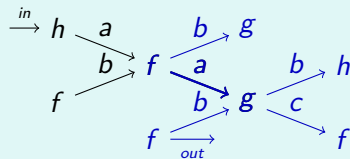
Birooted F -trees product

A rich algebraic structure:

Product = sync + fusion



(B · C)



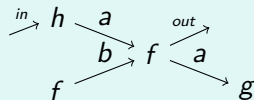
Theorem

Extended with 0 for the undefined case and 1 if needed, the resulting algebra is a *monoid* $\mathcal{B}(F)$.

An inverse semigroup

Definition (Inverses and left and right projections)

(B)



Lemma

B^{-1} is the unique element such that

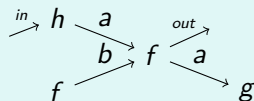
$$B \cdot B^{-1} \cdot B = B \text{ and } B^{-1} \cdot B \cdot B^{-1} = B^{-1}$$

i.e. $\mathcal{B}(F)$ is an inverse monoid [Law98].

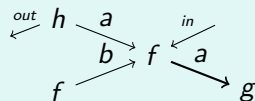
An inverse semigroup

Definition (Inverses and left and right projections)

(B)



(B^{-1})



Lemma

B^{-1} is the unique element such that

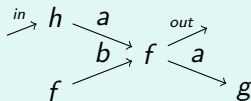
$$B \cdot B^{-1} \cdot B = B \text{ and } B^{-1} \cdot B \cdot B^{-1} = B^{-1}$$

i.e. $\mathcal{B}(F)$ is an inverse monoid [Law98].

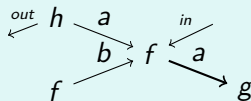
An inverse semigroup

Definition (Inverses and left and right projections)

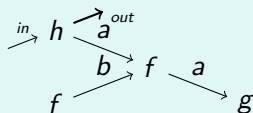
(B)



(B^{-1})



$\underbrace{(B \cdot B^{-1})}_{B^R}$



Lemma

B^{-1} is the unique element such that

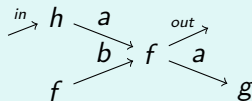
$$B \cdot B^{-1} \cdot B = B \text{ and } B^{-1} \cdot B \cdot B^{-1} = B^{-1}$$

i.e. $\mathcal{B}(F)$ is an inverse monoid [Law98].

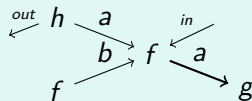
An inverse semigroup

Definition (Inverses and left and right projections)

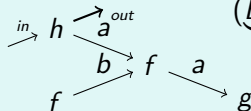
(B)



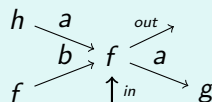
(B⁻¹)



$\underbrace{(B \cdot B^{-1})}_{B^R}$



$\underbrace{(B^{-1} \cdot B)}_{B^L}$



Lemma

B^{-1} is the unique element such that

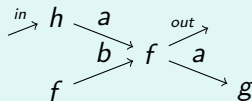
$$B \cdot B^{-1} \cdot B = B \text{ and } B^{-1} \cdot B \cdot B^{-1} = B^{-1}$$

i.e. $\mathcal{B}(F)$ is an inverse monoid [Law98].

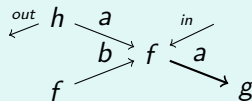
An inverse semigroup

Definition (Inverses and left and right projections)

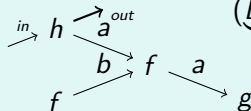
(B)



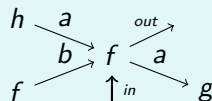
(B⁻¹)



$\underbrace{(B \cdot B^{-1})}_{B^R}$



$\underbrace{(B^{-1} \cdot B)}_{B^L}$



Lemma

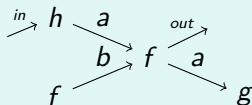
B^{-1} is the unique element such that

$$B \cdot B^{-1} \cdot B = B \text{ and } B^{-1} \cdot B \cdot B^{-1} = B^{-1}$$

i.e. $\mathcal{B}(F)$ is an inverse monoid [Law98].

The natural order

Sub birooted tree



Remark

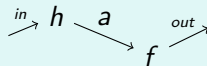
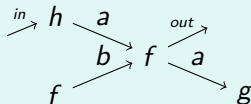
We have $B \leq C$ if and only if $B = B^R \cdot C$ (eq. $B = C \cdot B^L$).

Lemma

The idempotents are the subunits, i.e. $B \cdot B = B$ iff $B \leq 1$.

The natural order

Sub birooted tree



Remark

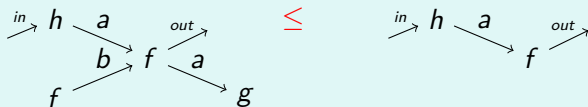
We have $B \leq C$ if and only if $B = B^R \cdot C$ (eq. $B = C \cdot B^L$).

Lemma

The idempotents are the subunits, i.e. $B \cdot B = B$ iff $B \leq 1$.

The natural order

Sub birooted tree to get **higher**



Remark

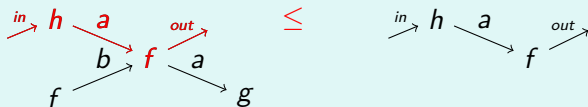
We have $B \leq C$ if and only if $B = B^R \cdot C$ (eq. $B = C \cdot B^L$).

Lemma

The idempotents are the subunits, i.e. $B \cdot B = B$ iff $B \leq 1$.

The natural order

Sub birooted tree to get **higher**



Remark

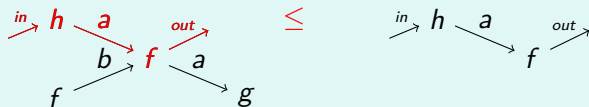
We have $B \leq C$ if and only if $B = B^R \cdot C$ (eq. $B = C \cdot B^L$).

Lemma

The idempotents are the subunits, i.e. $B \cdot B = B$ iff $B \leq 1$.

The natural order

Sub birooted tree to get **higher**



Remark

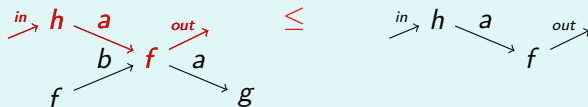
We have $B \leq C$ if and only if $B = B^R \cdot C$ (eq. $B = C \cdot B^L$).

Lemma

The idempotents are the subunits, i.e. $B \cdot B = B$ iff $B \leq 1$.

The natural order and the idempotents

Sub birooted tree to get **higher**



Remark

We have $B \leq C$ if and only if $B = B^R \cdot C$ (eq. $B = C \cdot B^L$).

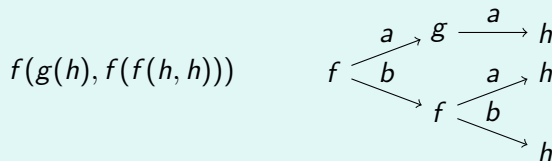
Lemma

The idempotents are the subunits, i.e. $B \cdot B = B$ iff $B \leq 1$.

Extending F -terms algebras

F -terms

With $\rho(f) = 2$, $\rho(g) = 1$ and $\rho(h) = 0$:



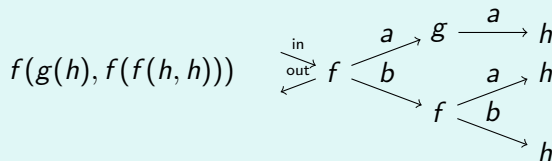
Observation

Classical (mono-rooted) F -trees can be **encoded** as idempotent birooted F -trees.

Extending F -terms algebras

F -terms

With $\rho(f) = 2$, $\rho(g) = 1$ and $\rho(h) = 0$:



Observation

Classical (mono-rooted) F -trees can be **encoded** as idempotent birooted F -trees.

Higher dimensional strings ?

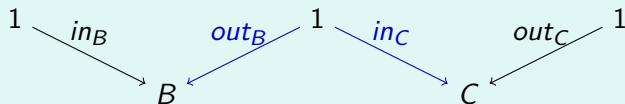
Remark (The product as a pushout)

Remark

- Represents all inverse semigroups (see [Ste90, Lee87]).
- Restricting to one-to-one morphisms (resp. plus 0) one gets E-unitary (resp. 0-E-unitary) inverse semigroups [Lee87].
- Construction also sketched in the proceedings [Jan14].

Higher dimensional strings ?

Remark (The product as a pushout)

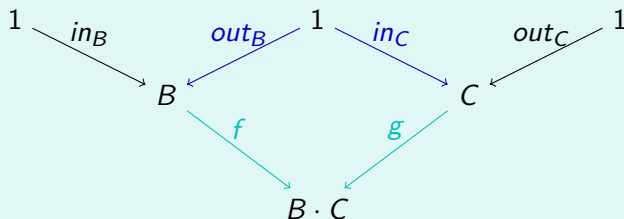


Remark

- Represents all inverse semigroups (see [Ste90, Lee87]).
- Restricting to one-to-one morphisms (resp. plus 0) one gets E-unitary (resp. 0-E-unitary) inverse semigroups [Lee87].
- Construction also sketched in the proceedings [Jan14].

Higher dimensional strings ?

Remark (The product as a pushout)

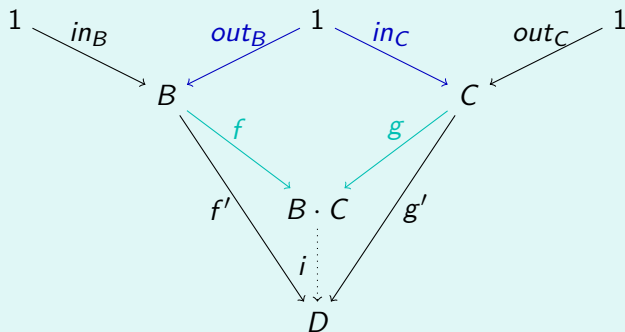


Remark

- Represents all inverse semigroups (see [Ste90, Lee87]).
- Restricting to one-to-one morphisms (resp. plus 0) one gets E-unitary (resp. 0-E-unitary) inverse semigroups [Lee87].
- Construction also sketched in the proceedings [Jan14].

Higher dimensional strings ?

Remark (The product as a pushout)

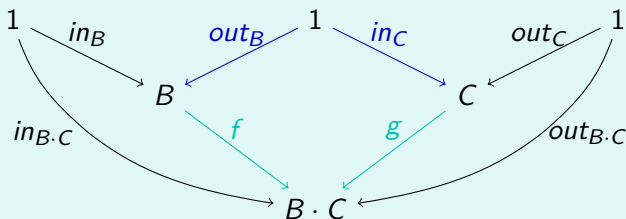


Remark

- Represents all inverse semigroups (see [Ste90, Lee87]).
- Restricting to one-to-one morphisms (resp. plus 0) one gets E-unitary (resp. 0-E-unitary) inverse semigroups [Lee87].
- Construction also sketched in the proceedings [Jan14].

Higher dimensional strings ?

Remark (The product as a pushout)

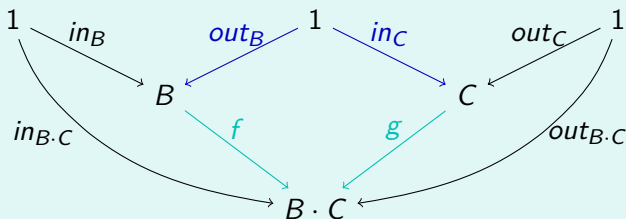


Remark

- Represents all inverse semigroups (see [Ste90, Lee87]).
- Restricting to one-to-one morphisms (resp. plus 0) one gets E-unitary (resp. 0-E-unitary) inverse semigroups [Lee87].
- Construction also sketched in the proceedings [Jan14].

Higher dimensional strings ?

Remark (The product as a pushout)



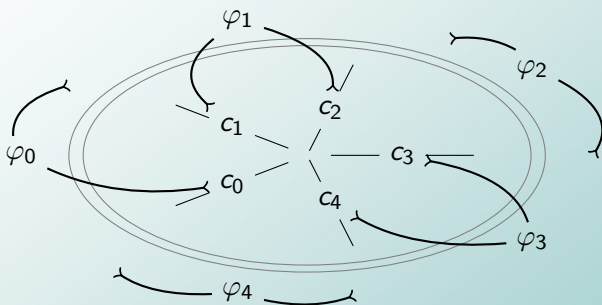
Remark

- Represents all inverse semigroups (see [Ste90, Lee87]).
- Restricting to one-to-one morphisms (resp. plus 0) one gets E-unitary (resp. 0-E-unitary) inverse semigroups [Lee87].
- Construction also sketched in the proceedings [Jan14].

3. Dining philosophy

More modeling experiments

The dining philosophers



Goal

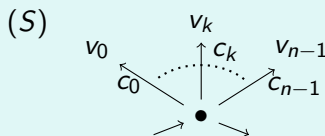
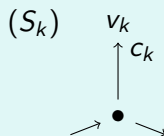
An incremental modeling of the dining philosopher problem and solution by means of birooted tree algebra.

Forks' local and global state modeling

Ingredients

Edge label: c_0, c_1, \dots, c_{n-1} , one per philosopher.

Vertex label: $\bullet, v \in V$ for some set V of local state values of forks.



States = Idempotent birooted trees

Local states S_k , global state S with

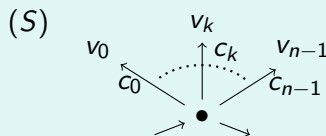
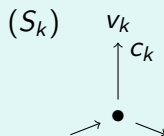
$$S \leq S_k \text{ for every } 0 \leq k < n, \text{ and } S = \prod_k S_k$$

Forks' local and global state modeling

Ingredients

Edge label: c_0, c_1, \dots, c_{n-1} , one per philosopher.

Vertex label: $\bullet, v \in V$ for some set V of local state values of forks.



States = Idempotent birooted trees

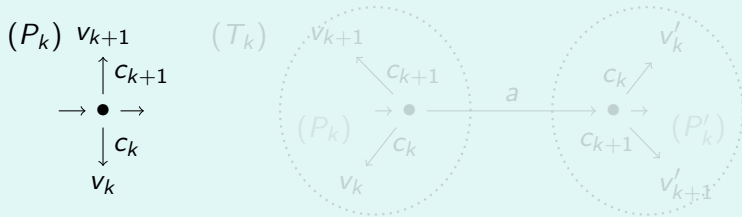
Local states S_k , global state S with

$$S \leq S_k \text{ for every } 0 \leq k < n, \text{ and } S = \prod_k S_k$$

Philosophers' local states and transitions

Philosopher's states

The state P_k of philosopher φ_k is of the form $P_k = S_k \cdot S_{k+1}$, i.e. built from the state of the two neighbor forks.



Philosopher's transition

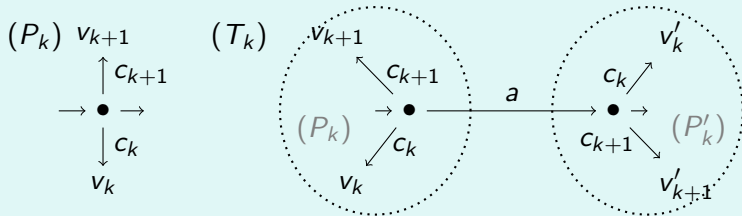
A transition T_k of the philosopher φ_k is built by relating two local state P_k and P'_k by some a -edge by $T_k = P_k \cdot a \cdot P'_k$ with

$$T_k^R \leq P_k, T_k^L \leq P'_k \text{ and } T_k = T_k^R \cdot a = a \cdot T_k^L = T_k^R \cdot a \cdot T_k^L$$

Philosophers' local states and transitions

Philosopher's states

The state P_k of philosopher φ_k is of the form $P_k = S_k \cdot S_{k+1}$, i.e. built from the state of the two neighbor forks.



Philosopher's transition

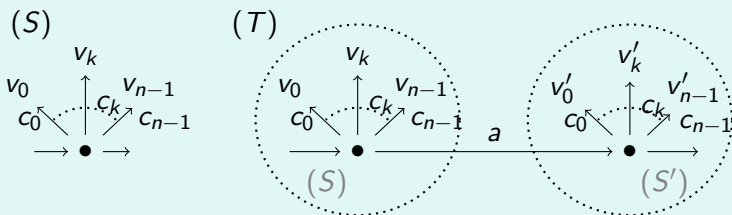
A transition T_k of the philosopher φ_k is built by relating two local state P_k and P'_k by some a -edge by $T_k = P_k \cdot a \cdot P'_k$ with

$$T_k^R \leq P_k, T_k^L \leq P'_k \text{ and } T_k = T_k^R \cdot a = a \cdot T_k^L = T_k^R \cdot a \cdot T_k^L$$

Philosophers' global states and transitions

Global transition

Of the form $T = S \cdot a \cdot S'$ with $T^R \leq S$ and $T^L \leq S'$.



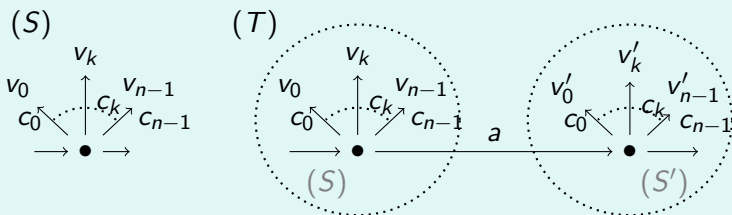
With local to global relationship

$$\begin{aligned}
 T &= \left(\prod_{0 \leq k < n} T_k^R \right) \cdot a = a \cdot \left(\prod_{0 \leq k < n} T_k^L \right) \\
 &= \left(\prod_{0 \leq k < n} T_k^R \right) \cdot a \cdot \left(\prod_{0 \leq k < n} T_k^L \right)
 \end{aligned}$$

Philosophers' global states and transitions

Global transition

Of the form $T = S \cdot a \cdot S'$ with $T^R \leq S$ and $T^L \leq S'$.



With local to global relationship

$$\begin{aligned}
 T &= \left(\prod_{0 \leq k < n} T_k^R \right) \cdot a = a \cdot \left(\prod_{0 \leq k < n} T_k^L \right) \\
 &= \left(\prod_{0 \leq k < n} T_k^R \right) \cdot a \cdot \left(\prod_{0 \leq k < n} T_k^L \right)
 \end{aligned}$$

Chandy and Misra's solution modeling

- a well chosen set V of fork local state values,
- the language L_k of local transitions of the form $T_k = P_k \cdot a \cdot P'_k$ that satisfies the algorithm, for every philosopher φ_k with $0 \leq k < n$,
- resulting global transitions defined by:

$$L = \prod_k (L_k)^R \cdot a = a \cdot \prod_k (L_k)^L = \prod_k (L_k)^R \cdot a \cdot \prod_k (L_k)^L$$

- starting from initial state S_0 , infinite global behaviors are given by

$$C = S_0 \cdot L^\omega$$

- check the algorithm is correct by analyzing the language C !!!

4. Languages

We need a simple but expressive notion of manageable languages of birooted \mathbf{F} -trees and, beyond, higher dimensional strings ?
What are the available language theoretic tools and concepts ?

Languages of birooted F -trees

Languages X and $Y \subseteq \mathcal{B}(F) - 0$ of *defined* birooted F -trees.

Operations on languages:

- sum : $X + Y = X \cup Y$,
- product : $X \cdot Y = \{B \cdot C : B \in X, C \in Y, B \cdot C \neq 0\}$,
- star : $X^* = \bigcup_{n \in \mathbb{N}} X^n$ with $X^0 = \{1\}$ and $X^{n+1} = X \cdot X^n$,

and, thanks to the inverse monoid structure of $\mathcal{B}(F)$:

- inverse : $X^{-1} = \{B^{-1} : B \in X\}$,
- idempotent projection : $X^E = \{B : B \in X, B \cdot B = B\}$,
- left and right projection : $X^L = \{B^{-1} \cdot B : B \in X\}$ and $X^R = \{B \cdot B^{-1} : B \in X\}$,
- up and down closures : $X^\uparrow = \{B \in \mathcal{B}(F) : \exists C \in X, C \leq B\}$
and $X^\downarrow = \{B \in \mathcal{B}(F) - 0 : \exists C \in X, B \leq C\}$.

Languages of birooted F -trees

Languages X and $Y \subseteq \mathcal{B}(F) - 0$ of *defined* birooted F -trees.

Operations on languages:

- **sum** : $X + Y = X \cup Y$,
- **product** : $X \cdot Y = \{B \cdot C : B \in X, C \in Y, B \cdot C \neq 0\}$,
- **star** : $X^* = \bigcup_{n \in \mathbb{N}} X^n$ with $X^0 = \{1\}$ and $X^{n+1} = X \cdot X^n$,

and, thanks to the inverse monoid structure of $\mathcal{B}(F)$:

- **inverse** : $X^{-1} = \{B^{-1} : B \in X\}$,
- **idempotent projection** : $X^E = \{B : B \in X, B \cdot B = B\}$,
- **left and right projection** : $X^L = \{B^{-1} \cdot B : B \in X\}$ and $X^R = \{B \cdot B^{-1} : B \in X\}$,
- **up and down closures** : $X^\uparrow = \{B \in \mathcal{B}(F) : \exists C \in X, C \leq B\}$
and $X^\downarrow = \{B \in \mathcal{B}(F) - 0 : \exists C \in X, B \leq C\}$.

Languages of birooted F -trees

Languages X and $Y \subseteq \mathcal{B}(F) - 0$ of *defined* birooted F -trees.

Operations on languages:

- **sum** : $X + Y = X \cup Y$,
- **product** : $X \cdot Y = \{B \cdot C : B \in X, C \in Y, B \cdot C \neq 0\}$,
- **star** : $X^* = \bigcup_{n \in \mathbb{N}} X^n$ with $X^0 = \{1\}$ and $X^{n+1} = X \cdot X^n$,

and, thanks to the inverse monoid structure of $\mathcal{B}(F)$:

- **inverse** : $X^{-1} = \{B^{-1} : B \in X\}$,
- **idempotent projection** : $X^E = \{B : B \in X, B \cdot B = B\}$,
- **left and right projection** : $X^L = \{B^{-1} \cdot B : B \in X\}$ and $X^R = \{B \cdot B^{-1} : B \in X\}$,
- **up and down closures** : $X^\uparrow = \{B \in \mathcal{B}(F) : \exists C \in X, C \leq B\}$
and $X^\downarrow = \{B \in \mathcal{B}(F) - 0 : \exists C \in X, B \leq C\}$.

Languages of birooted F -trees

Languages X and $Y \subseteq \mathcal{B}(F) - 0$ of *defined* birooted F -trees.

Operations on languages:

- **sum** : $X + Y = X \cup Y$,
- **product** : $X \cdot Y = \{B \cdot C : B \in X, C \in Y, B \cdot C \neq 0\}$,
- **star** : $X^* = \bigcup_{n \in \mathbb{N}} X^n$ with $X^0 = \{1\}$ and $X^{n+1} = X \cdot X^n$,

and, thanks to the inverse monoid structure of $\mathcal{B}(F)$:

- **inverse** : $X^{-1} = \{B^{-1} : B \in X\}$,
- **idempotent projection** : $X^E = \{B : B \in X, B \cdot B = B\}$,
- **left and right projection** : $X^L = \{B^{-1} \cdot B : B \in X\}$ and $X^R = \{B \cdot B^{-1} : B \in X\}$,
- **up and down closures** : $X^\uparrow = \{B \in \mathcal{B}(F) : \exists C \in X, C \leq B\}$ and $X^\downarrow = \{B \in \mathcal{B}(F) - 0 : \exists C \in X, B \leq C\}$.

Languages of birooted F -trees

Languages X and $Y \subseteq \mathcal{B}(F) - 0$ of *defined* birooted F -trees.

Operations on languages:

- **sum** : $X + Y = X \cup Y$,
- **product** : $X \cdot Y = \{B \cdot C : B \in X, C \in Y, B \cdot C \neq 0\}$,
- **star** : $X^* = \bigcup_{n \in \mathbb{N}} X^n$ with $X^0 = \{1\}$ and $X^{n+1} = X \cdot X^n$,

and, thanks to the inverse monoid structure of $\mathcal{B}(F)$:

- **inverse** : $X^{-1} = \{B^{-1} : B \in X\}$,
- **idempotent projection** : $X^E = \{B : B \in X, B \cdot B = B\}$,
- **left and right projection** : $X^L = \{B^{-1} \cdot B : B \in X\}$ and $X^R = \{B \cdot B^{-1} : B \in X\}$,
- **up and down closures** : $X^\uparrow = \{B \in \mathcal{B}(F) : \exists C \in X, C \leq B\}$ and $X^\downarrow = \{B \in \mathcal{B}(F) - 0 : \exists C \in X, B \leq C\}$.

Languages of birooted F -trees

Languages X and $Y \subseteq \mathcal{B}(F) - 0$ of *defined* birooted F -trees.

Operations on languages:

- **sum** : $X + Y = X \cup Y$,
- **product** : $X \cdot Y = \{B \cdot C : B \in X, C \in Y, B \cdot C \neq 0\}$,
- **star** : $X^* = \bigcup_{n \in \mathbb{N}} X^n$ with $X^0 = \{1\}$ and $X^{n+1} = X \cdot X^n$,

and, thanks to the inverse monoid structure of $\mathcal{B}(F)$:

- **inverse** : $X^{-1} = \{B^{-1} : B \in X\}$,
- **idempotent projection** : $X^E = \{B : B \in X, B \cdot B = B\}$,
- **left and right projection** : $X^L = \{B^{-1} \cdot B : B \in X\}$ and $X^R = \{B \cdot B^{-1} : B \in X\}$,
- **up and down closures** : $X^\uparrow = \{B \in \mathcal{B}(F) : \exists C \in X, C \leq B\}$ and $X^\downarrow = \{B \in \mathcal{B}(F) - 0 : \exists C \in X, B \leq C\}$.

Languages of birooted F -trees

Languages X and $Y \subseteq \mathcal{B}(F) - 0$ of *defined* birooted F -trees.

Operations on languages:

- **sum** : $X + Y = X \cup Y$,
- **product** : $X \cdot Y = \{B \cdot C : B \in X, C \in Y, B \cdot C \neq 0\}$,
- **star** : $X^* = \bigcup_{n \in \mathbb{N}} X^n$ with $X^0 = \{1\}$ and $X^{n+1} = X \cdot X^n$,

and, thanks to the inverse monoid structure of $\mathcal{B}(F)$:

- **inverse** : $X^{-1} = \{B^{-1} : B \in X\}$,
- **idempotent projection** : $X^E = \{B : B \in X, B \cdot B = B\}$,
- **left and right projection** : $X^L = \{B^{-1} \cdot B : B \in X\}$ and $X^R = \{B \cdot B^{-1} : B \in X\}$,
- **up and down closures** : $X^\uparrow = \{B \in \mathcal{B}(F) : \exists C \in X, C \leq B\}$ and $X^\downarrow = \{B \in \mathcal{B}(F) - 0 : \exists C \in X, B \leq C\}$.

Languages of birooted F -trees

Languages X and $Y \subseteq \mathcal{B}(F) - 0$ of *defined* birooted F -trees.

Operations on languages:

- **sum** : $X + Y = X \cup Y$,
- **product** : $X \cdot Y = \{B \cdot C : B \in X, C \in Y, B \cdot C \neq 0\}$,
- **star** : $X^* = \bigcup_{n \in \mathbb{N}} X^n$ with $X^0 = \{1\}$ and $X^{n+1} = X \cdot X^n$,

and, thanks to the inverse monoid structure of $\mathcal{B}(F)$:

- **inverse** : $X^{-1} = \{B^{-1} : B \in X\}$,
- **idempotent projection** : $X^E = \{B : B \in X, B \cdot B = B\}$,
- **left and right projection** : $X^L = \{B^{-1} \cdot B : B \in X\}$ and $X^R = \{B \cdot B^{-1} : B \in X\}$,
- **up and down closures** : $X^\uparrow = \{B \in \mathcal{B}(F) : \exists C \in X, C \leq B\}$
and $X^\downarrow = \{B \in \mathcal{B}(F) - 0 : \exists C \in X, B \leq C\}$.

The known case of word languages

Definability classes of languages of words

A language $L \subseteq A^*$ is

- **REC** when $L = \varphi^{-1}(\varphi(L))$ for some morphism $\varphi : A^* \rightarrow S$ with finite monoid S ,
- **REG** when L is definable by a regular expression, i.e. definable from finite languages combined with sum, product and star,
- **MSO** when L is definable in Monadic Second Order Logic.

Theorem (Kleene, Rabin, Scott, Buchi, etc. . .)

Over words, $\underbrace{REC = REG = MSO}_{\text{finite automata}}$

The known case of word languages

Definability classes of languages of words

A language $L \subseteq A^*$ is

- **REC** when $L = \varphi^{-1}(\varphi(L))$ for some **morphism** $\varphi : A^* \rightarrow S$ with **finite monoid** S ,
- **REG** when L is definable by a **regular expression**, i.e. definable from finite languages combined with sum, product and star,
- **MSO** when L is definable in **Monadic Second Order Logic**.

Theorem (Kleene, Rabin, Scott, Buchi, etc. . .)

Over words, $\underbrace{REC = REG = MSO}_{\text{finite automata}}$

The known case of word languages

Definability classes of languages of words

A language $L \subseteq A^*$ is

- **REC** when $L = \varphi^{-1}(\varphi(L))$ for some **morphism** $\varphi : A^* \rightarrow S$ with **finite monoid** S ,
- **REG** when L is definable by a **regular expression**, i.e. definable from finite languages combined with sum, product and star,
- **MSO** when L is definable in **Monadic Second Order Logic**.

Theorem (Kleene, Rabin, Scott, Buchi, etc. . .)

Over words, $\underbrace{REC = REG = MSO}_{\text{finite automata}}$

The known case of word languages

Definability classes of languages of words

A language $L \subseteq A^*$ is

- **REC** when $L = \varphi^{-1}(\varphi(L))$ for some **morphism** $\varphi : A^* \rightarrow S$ with **finite monoid** S ,
- **REG** when L is definable by a **regular expression**, i.e. definable from finite languages combined with sum, product and star,
- **MSO** when L is definable in **Monadic Second Order Logic**.

Theorem (Kleene, Rabin, Scott, Buchi, etc. . .)

Over words, $\underbrace{REC = REG = MSO}_{\text{finite automata}}$

The known case of word languages

Definability classes of languages of words

A language $L \subseteq A^*$ is

- **REC** when $L = \varphi^{-1}(\varphi(L))$ for some **morphism** $\varphi : A^* \rightarrow S$ with **finite monoid** S ,
- **REG** when L is definable by a **regular expression**, i.e. definable from finite languages combined with sum, product and star,
- **MSO** when L is definable in **Monadic Second Order Logic**.

Theorem (Kleene, Rabin, Scott, Buchi, etc. . .)

Over words, $\underbrace{REC = REG = MSO}_{\text{finite automata}}$

The new case of birooted F -tree languages

Classes of languages of F -trees

A language $X \subseteq \mathcal{B}(F)$ is

- **REC** when $X = \varphi^{-1}(\varphi(X))$ for some morphism $\varphi : A^* \rightarrow S$ with finite monoid S ,
- **k -REG** when it is definable by a regular expression extended by idempotent projection with nesting depth at most k ,
- **MSO** when it is MSO and upward closed in the natural order.

Theorem (Robustness)

The class MSO is closed under boolean, product, star, inverse, projections, upward and downward closure. . .

Theorem (Expressiveness)

Over birooted F -trees,

$$REC \subset REG \subset 1\text{-REG} \subseteq \dots \subseteq k\text{-REG} \subseteq \dots \subseteq MSO$$

The new case of birooted F -tree languages

Classes of languages of F -trees

A language $X \subseteq \mathcal{B}(F)$ is

- **REC** when $X = \varphi^{-1}(\varphi(X))$ for some **morphism** $\varphi : A^* \rightarrow S$ with **finite monoid** S ,
- **k -REG** when it is definable by a regular expression extended by idempotent projection with nesting depth at most k ,
- **MSO** when it is MSO and **upward closed** in the natural order.

Theorem (Robustness)

The class MSO is closed under boolean, product, star, inverse, projections, upward and downward closure. . .

Theorem (Expressiveness)

Over birooted F -trees,

$$REC \subset REG \subset 1\text{-REG} \subseteq \dots \subseteq k\text{-REG} \subseteq \dots \subseteq MSO$$

The new case of birooted F -tree languages

Classes of languages of F -trees

A language $X \subseteq \mathcal{B}(F)$ is

- **REC** when $X = \varphi^{-1}(\varphi(X))$ for some **morphism** $\varphi : A^* \rightarrow S$ with **finite monoid** S ,
- **k -REG** when it is definable by a regular expression extended by **idempotent projection with nesting depth at most k** ,
- **MSO** when it is MSO and **upward closed** in the natural order.

Theorem (Robustness)

The class MSO is closed under boolean, product, star, inverse, projections, upward and downward closure. . .

Theorem (Expressiveness)

Over birooted F -trees,

$$REC \subset REG \subset 1\text{-REG} \subseteq \dots \subseteq k\text{-REG} \subseteq \dots \subseteq MSO$$

The new case of birooted F -tree languages

Classes of languages of F -trees

A language $X \subseteq \mathcal{B}(F)$ is

- **REC** when $X = \varphi^{-1}(\varphi(X))$ for some **morphism** $\varphi : A^* \rightarrow S$ with **finite monoid** S ,
- **k -REG** when it is definable by a regular expression extended by **idempotent projection with nesting depth at most k** ,
- **MSO** when it is MSO and **upward closed** in the natural order.

Theorem (Robustness)

The class MSO is closed under boolean, product, star, inverse, projections, upward and downward closure. . .

Theorem (Expressiveness)

Over birooted F -trees,

$$REC \subset REG \subset 1\text{-REG} \subseteq \dots \subseteq k\text{-REG} \subseteq \dots \subseteq MSO$$

The new case of birooted F -tree languages

Classes of languages of F -trees

A language $X \subseteq \mathcal{B}(F)$ is

- **REC** when $X = \varphi^{-1}(\varphi(X))$ for some **morphism** $\varphi : A^* \rightarrow S$ with **finite monoid** S ,
- **k -REG** when it is definable by a regular expression extended by **idempotent projection with nesting depth at most k** ,
- **MSO** when it is MSO and **upward closed** in the natural order.

Theorem (Robustness)

The class MSO is closed under boolean, product, star, inverse, projections, upward and downward closure. . .

Theorem (Expressiveness)

Over birooted F -trees,

$$\text{REC} \subset \text{REG} \subset 1\text{-REG} \subseteq \dots \subseteq k\text{-REG} \subseteq \dots \subseteq \text{MSO}$$

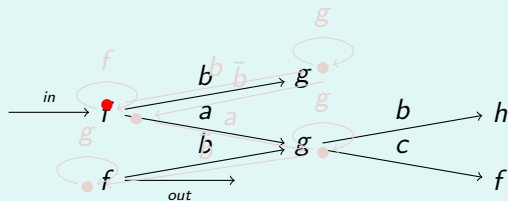
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

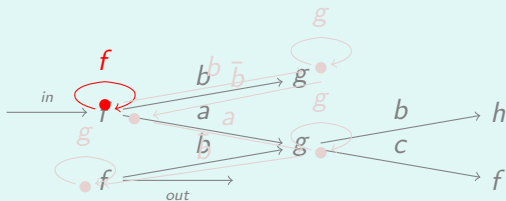
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

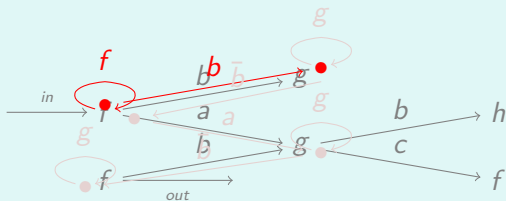
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

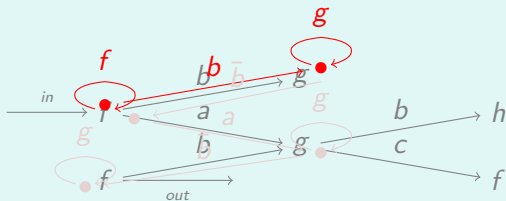
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

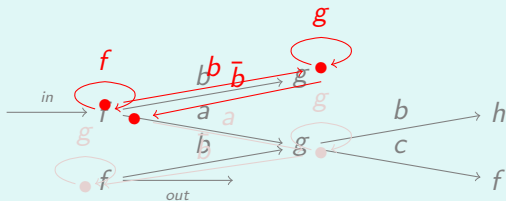
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

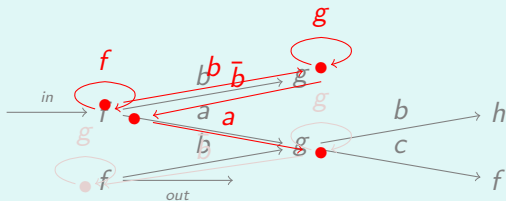
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

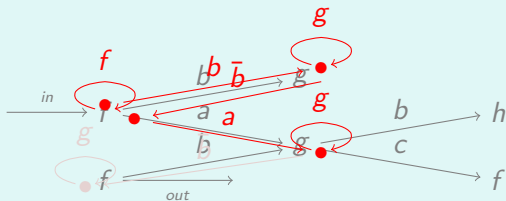
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

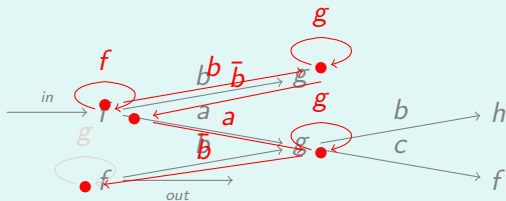
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

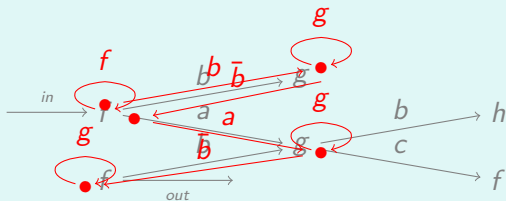
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

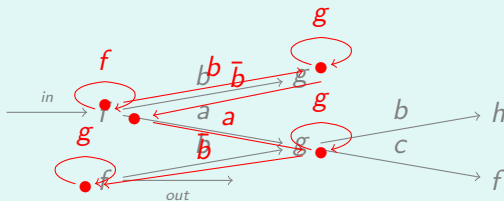
Walking automata and downward closed languages

Definition (Walking automata – see e.g. [Boj08])

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

that read partial traversals of birooted trees.

Example of a walking run



Remark

Walking runs are preserved downward in the natural order.

Walking automata and downward closed languages

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

Definition (Recognized language: $L(\mathcal{A})$)

The set of $B \in \mathcal{B}(F)$ for which there is an **accepting walking run**, from the **input root in initial state q_0** to the **output root in an accepting state $q \in T$** , possibly with invisible pebble mechanisms.

Theorem (The walking hierarchy [DJ13, Jan13d])

$$REC^\downarrow \subset REG^\downarrow \subset 1\text{-}REG^\downarrow \subseteq \dots \subseteq k\text{-}REG^\downarrow \subseteq \dots \subseteq MSO^\downarrow$$

- REC^\downarrow by strongly deterministic walking automata,
- REG^\downarrow by non deterministic walking automata,
- $k\text{-}REG^\downarrow$ by k -(invisible)-pebble walking automata,
- MSO^\downarrow by ω -(invisible)-pebble walking automata.

Walking automata and downward closed languages

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

Definition (Recognized language: $L(\mathcal{A})$)

The set of $B \in \mathcal{B}(F)$ for which there is an **accepting walking run**, from the **input root in initial state q_0** to the **output root in an accepting state $q \in T$** , possibly with invisible pebble mechanisms.

Theorem (The walking hierarchy [DJ13, Jan13d])

$$REC^\downarrow \subset REG^\downarrow \subset 1-REG^\downarrow \subseteq \dots \subseteq k-REG^\downarrow \subseteq \dots \subseteq MSO^\downarrow$$

- REC^\downarrow by *strongly deterministic walking automata*,
- REG^\downarrow by *non deterministic walking automata*,
- $k-REG^\downarrow$ by *k -(invisible)-pebble walking automata*,
- MSO^\downarrow by *ω -(invisible)-pebble walking automata*.

Walking automata and downward closed languages

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

Definition (Recognized language: $L(\mathcal{A})$)

The set of $B \in \mathcal{B}(F)$ for which there is an **accepting walking run**, from the **input root in initial state q_0** to the **output root in an accepting state $q \in T$** , possibly with invisible pebble mechanisms.

Theorem (The walking hierarchy [DJ13, Jan13d])

$$REC^\downarrow \subset REG^\downarrow \subset 1-REG^\downarrow \subseteq \dots \subseteq k-REG^\downarrow \subseteq \dots \subseteq MSO^\downarrow$$

- REC^\downarrow by *strongly deterministic walking automata*,
- REG^\downarrow by *non deterministic walking automata*,
- $k-REG^\downarrow$ by *k -(invisible)-pebble walking automata*,
- MSO^\downarrow by *ω -(invisible)-pebble walking automata*.

Walking automata and downward closed languages

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

Definition (Recognized language: $L(\mathcal{A})$)

The set of $B \in \mathcal{B}(F)$ for which there is an **accepting walking run**, from the **input root in initial state q_0** to the **output root in an accepting state $q \in T$** , possibly with invisible pebble mechanisms.

Theorem (The walking hierarchy [DJ13, Jan13d])

$$REC^\downarrow \subset REG^\downarrow \subset 1-REG^\downarrow \subseteq \dots \subseteq k-REG^\downarrow \subseteq \dots \subseteq MSO^\downarrow$$

- REC^\downarrow by *strongly deterministic walking automata*,
- REG^\downarrow by *non deterministic walking automata*,
- $k-REG^\downarrow$ by *k -(invisible)-pebble walking automata*,
- MSO^\downarrow by *ω -(invisible)-pebble walking automata*.

Walking automata and downward closed languages

$$\mathcal{A} = \langle Q, q_0, T, \delta : (A + \bar{A} + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

Definition (Recognized language: $L(\mathcal{A})$)

The set of $B \in \mathcal{B}(F)$ for which there is an **accepting walking run**, from the **input root in initial state q_0** to the **output root in an accepting state $q \in T$** , possibly with invisible pebble mechanisms.

Theorem (The walking hierarchy [DJ13, Jan13d])

$$REC^\downarrow \subset REG^\downarrow \subset 1-REG^\downarrow \subseteq \dots \subseteq k-REG^\downarrow \subseteq \dots \subseteq MSO^\downarrow$$

- REC^\downarrow by *strongly deterministic walking automata*,
- REG^\downarrow by *non deterministic walking automata*,
- $k-REG^\downarrow$ by *k -(invisible)-pebble walking automata*,
- MSO^\downarrow by *ω -(invisible)-pebble walking automata*.

Non det. automata and downward closed languages

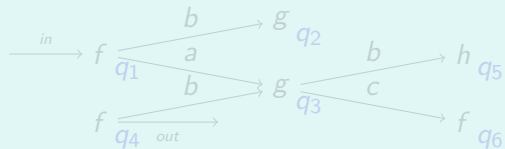
Definition (Non deterministic automata – see e.g. [Tho97])

$$\mathcal{A} = \langle Q, W, \delta : (A + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

with run that are markings $\rho : \text{dom}(B) \rightarrow Q$ such that:

- **Vertex coherence:** for all $v \in \text{dom}(B)$ labeled by $f \in F$,
 $(\rho(v), \rho(v)) \in \delta(f)$
- **Edge coherence:** for all edge $v \xrightarrow{a} w$ in B ,
 $(\rho(v), \rho(w)) \in \delta(a)$

A run in picture



Coherence ex.:

$$(q_2, q_2) \in \delta(g)$$

$$(q_4, q_3) \in \delta(b)$$

Non det. automata and downward closed languages

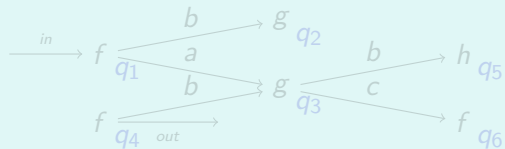
Definition (Non deterministic automata – see e.g. [Tho97])

$$\mathcal{A} = \langle Q, W, \delta : (A + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

with run that are markings $\rho : \text{dom}(B) \rightarrow Q$ such that:

- **Vertex coherence:** for all $v \in \text{dom}(B)$ labeled by $f \in F$,
 $(\rho(v), \rho(v)) \in \delta(f)$
- **Edge coherence:** for all edge $v \xrightarrow{a} w$ in B ,
 $(\rho(v), \rho(w)) \in \delta(a)$

A run in picture



Coherence ex.:

$$(q_2, q_2) \in \delta(g)$$

$$(q_4, q_3) \in \delta(b)$$

Non det. automata and downward closed languages

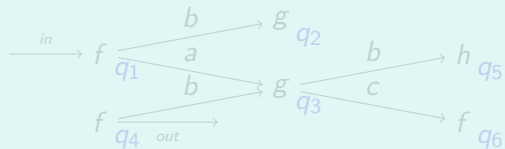
Definition (Non deterministic automata – see e.g. [Tho97])

$$\mathcal{A} = \langle Q, W, \delta : (A + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

with run that are markings $\rho : \text{dom}(B) \rightarrow Q$ such that:

- **Vertex coherence:** for all $v \in \text{dom}(B)$ labeled by $f \in F$,
 $(\rho(v), \rho(v)) \in \delta(f)$
- **Edge coherence:** for all edge $v \xrightarrow{a} w$ in B ,
 $(\rho(v), \rho(w)) \in \delta(a)$

A run in picture



Coherence ex.:

$$(q_2, q_2) \in \delta(g)$$

$$(q_4, q_3) \in \delta(b)$$

Non det. automata and downward closed languages

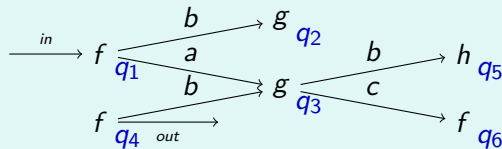
Definition (Non deterministic automata – see e.g. [Tho97])

$$\mathcal{A} = \langle Q, W, \delta : (A + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

with run that are markings $\rho : \text{dom}(B) \rightarrow Q$ such that:

- **Vertex coherence:** for all $v \in \text{dom}(B)$ labeled by $f \in F$,
 $(\rho(v), \rho(v)) \in \delta(f)$
- **Edge coherence:** for all edge $v \xrightarrow{a} w$ in B ,
 $(\rho(v), \rho(w)) \in \delta(a)$

A run in picture



Coherence ex.:

$$(q_2, q_2) \in \delta(g)$$

$$(q_4, q_3) \in \delta(b)$$

Non det. automata and downward closed languages

$$\mathcal{A} = \langle Q, W, \delta : (A + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

Definition (Recognized language: $L(\mathcal{A})$)

The set of $B \in \mathcal{B}(F)$ for which there is an **marking run** ρ that satisfies the **acceptance condition** $(\rho(in_B), \rho(out_B)) \in W$.

Remark

If $X \subseteq \mathcal{B}(F) - 0$ is recognized by a (finite) automaton then X is upward closed w.r.t. the natural order.

Theorem (Expressiveness [Jan13c, Jan13a])

A language $X \subseteq \mathcal{B}(F) - 0$ is recognized by a non deterministic finite state automaton \mathcal{A} if and only if $X \in MSO^\uparrow$.

Non det. automata and downward closed languages

$$\mathcal{A} = \langle Q, W, \delta : (A + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

Definition (Recognized language: $L(\mathcal{A})$)

The set of $B \in \mathcal{B}(F)$ for which there is an **marking run** ρ that satisfies the **acceptance condition** $(\rho(in_B), \rho(out_B)) \in W$.

Remark

If $X \subseteq \mathcal{B}(F) - 0$ is recognized by a (finite) automaton then X is **upward closed** w.r.t. the natural order.

Theorem (Expressiveness [Jan13c, Jan13a])

A language $X \subseteq \mathcal{B}(F) - 0$ is recognized by a non deterministic finite state automaton \mathcal{A} if and only if $X \in MSO^\uparrow$.

Non det. automata and downward closed languages

$$\mathcal{A} = \langle Q, W, \delta : (A + F) \rightarrow \mathcal{P}(Q \times Q) \rangle$$

Definition (Recognized language: $L(\mathcal{A})$)

The set of $B \in \mathcal{B}(F)$ for which there is an **marking run** ρ that satisfies the **acceptance condition** $(\rho(in_B), \rho(out_B)) \in W$.

Remark

If $X \subseteq \mathcal{B}(F) - 0$ is recognized by a (finite) automaton then X is **upward closed** w.r.t. the natural order.

Theorem (Expressiveness [Jan13c, Jan13a])

A language $X \subseteq \mathcal{B}(F) - 0$ is recognized by a non deterministic finite state automaton \mathcal{A} if and only if $X \in \text{MSO}^\uparrow$.

Extension to higher dimensional strings

Question

- What is the behavior of walking automata on more complex structures ?
- Idem for non deterministic automata ?
- Relationship with graph acceptors ($\exists MSO$) ?

5. Quasi-recognizability

Quasi-inverse monoids and the (algebraic) boolean closure of FSA.

Downward closed languages of an inverse monoid

Definition

Let S be an **inverse monoid** with natural order \leq . Let $\mathcal{P}^\downarrow(S)$ be the set of its **downward closed non empty subsets** of S with the point-wise extension of the product.

Lemma

Then $\mathcal{P}^\downarrow(S)$ ordered by inclusion is a ordered monoid with:

- **unit:** $X \cdot U(S) = X = U(S) \cdot X$ with $U(S) = \{z \in S : z \leq 1\}$,
- **stable order:** for every Z if $X \subseteq Y$ then $Z \cdot X \subseteq Z \cdot Y$ and $X \cdot Z \subseteq Y \cdot Z$,
- **idempotent subunits:** if $X \subseteq U(S)$ then $X \cdot X = X$,
- **left and right local units:** for every $X \in \mathcal{P}^\downarrow(S)$,
 - $X^R = \{x^R \leq 1 : x \in X\} = \bigcap \{Z \subseteq U(S) : X \cdot Z = X\}$,
 - $X^L = \{x^L \leq 1 : x \in X\} = \bigcap \{Z \subseteq U(S) : Z \cdot X = X\}$.

But, it is not necessarily inverse !

Quasi-inverse monoids

Generalizing the previous properties to ordered monoids:

Definition

A ordered monoid $\langle S, \cdot, 1, \leq \rangle$ is **adequately ordered** when:

- **Stable order:** if $x \leq y$ then $z \cdot x \leq z \cdot y$ and $x \cdot z \leq y \cdot z$,
- **Idempotent subunits:** if $x \leq 1$ then $x \cdot x = x$,
- **Left and right projections:** for every $x \in S$, both projections

$$\underbrace{x^L = \min\{z \leq 1 : x \cdot z = x\}}_{\text{behaves "like" } x^{-1}x} \quad \text{and} \quad \underbrace{x^R = \min\{z \leq 1 : z \cdot x = x\}}_{\text{behaves "like" } xx^{-1}}$$

exist.

Examples of quasi-inverse monoids

Examples

- Trivially ordered monoids with $x^L = x^R = 1$.
- Inverse monoids with $x^L = x^{-1}x$ and $x^R = xx^{-1}$.
- Finite partially ordered monoid with idempotent subunits,

Remark

Such a definition is strongly related [Jan12a] with the studies of semigroups with local units developed by the “York School” [Fou77] with, in particular, U -semiadequate and Ehresmann semigroups [Law91].

Examples of quasi-inverse monoids

Examples

- Trivially ordered monoids with $x^L = x^R = 1$.
- Inverse monoids with $x^L = x^{-1}x$ and $x^R = xx^{-1}$.
- Finite partially ordered monoid with idempotent subunits,

Remark

Such a definition is strongly related [Jan12a] with the studies of semigroups with local units developed by the “York School” [Fou77] with, in particular, U -semiadequate and Ehresmann semigroups [Law91].

Examples of quasi-inverse monoids

Examples

- Trivially ordered monoids with $x^L = x^R = 1$.
- Inverse monoids with $x^L = x^{-1}x$ and $x^R = xx^{-1}$.
- Finite partially ordered monoid with idempotent subunits,

Remark

Such a definition is strongly related [Jan12a] with the studies of semigroups with local units developed by the “York School” [Fou77] with, in particular, U -semiadequate and Ehresmann semigroups [Law91].

Examples of quasi-inverse monoids

Examples

- Trivially ordered monoids with $x^L = x^R = 1$.
- Inverse monoids with $x^L = x^{-1}x$ and $x^R = xx^{-1}$.
- Finite partially ordered monoid with idempotent subunits,

Remark

Such a definition is strongly related [Jan12a] with the studies of semigroups with local units developed by the “York School” [Fou77] with, in particular, U -semiadequate and Ehresmann semigroups [Law91].

Runs of non deterministic automata revisited

Let $\mathcal{A} = \langle Q, \delta, W \rangle$ be a non det. automaton and let

$$\varphi : \mathcal{B}(F) \rightarrow \mathcal{P}(Q \times Q)$$

be the mapping defined by $\varphi(0) = \emptyset$, $\varphi(1) = I_Q$ and, for every non trivial F -tree B , the set $\varphi(B)$ defined as the set of pairs $(\rho(in_B), \rho(out_B))$ for runs ρ of \mathcal{A} over B .

Lemma

The mapping φ recognizes $L(\mathcal{A})$ in the sense that

$$L(\mathcal{A}) = \varphi^{-1}(\varphi(L(\mathcal{A})))$$

Runs of non deterministic automata revisited

Let $\mathcal{A} = \langle Q, \delta, W \rangle$ be a non det. automaton and let

$$\varphi : \mathcal{B}(F) \rightarrow \mathcal{P}(Q \times Q)$$

be the mapping defined by $\varphi(0) = \emptyset$, $\varphi(1) = I_Q$ and, for every non trivial F -tree B , the set $\varphi(B)$ defined as the set of pairs $(\rho(in_B), \rho(out_B))$ for runs ρ of \mathcal{A} over B .

Lemma

The mapping φ recognizes $L(\mathcal{A})$ in the sense that

$$L(\mathcal{A}) = \varphi^{-1}(\varphi(L(\mathcal{A})))$$

Runs of non deterministic automata revisited

Let $\mathcal{A} = \langle Q, \delta, W \rangle$ be a non det. automaton and let

$$\varphi : \mathcal{B}(F) \rightarrow \mathcal{P}(Q \times Q)$$

be the mapping defined by $\varphi(0) = \emptyset$, $\varphi(1) = I_Q$ and, for every non trivial F -tree B , the set $\varphi(B)$ defined as the set of pairs $(\rho(in_B), \rho(out_B))$ for runs ρ of \mathcal{A} over B .

Lemma

The mapping φ recognizes $L(\mathcal{A})$ in the sense that

$$L(\mathcal{A}) = \varphi^{-1}(\varphi(L(\mathcal{A})))$$

Effectivity

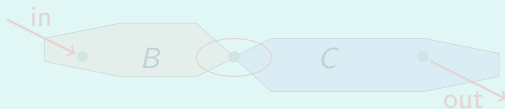
Lemma

Let $B \in \mathcal{B}(F) - 0$. Then $\varphi(B)$ is definable in MSO and computable in linear time.

Proof.

The mapping φ preserves:

- left and right projections,
- disjoint products (written $B * C$),



and every birooted F -tree is definable as a linear size combination of elementary trees with disjoint products and left and right projections. □

Effectivity

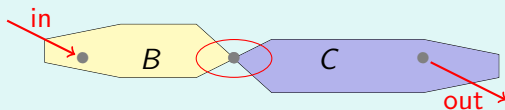
Lemma

Let $B \in \mathcal{B}(F) - 0$. Then $\varphi(B)$ is definable in MSO and computable in linear time.

Proof.

The mapping φ preserves:

- left and right projections,
- disjoint products (written $B * C$),



and every birooted F -tree is definable as a **linear size combination** of elementary trees with disjoint products and left and right projections. □

Towards quasi-recognizability

Definition (Adequate premorphism)

A mapping $\varphi : \mathcal{B}(F) \rightarrow S$ with S quasi-inverse such that:

- unit: $\varphi(1) = 1$,
- monotonic: if $B \leq C$ then $\varphi(B) \leq \varphi(C)$,
- disjoint product: if $\exists B * C$ then $\varphi(B * C) = \varphi(B) \cdot \varphi(C)$,
- projections: $\varphi(B^L) = (\varphi(B))^L$ and $\varphi(B^R) = (\varphi(B))^R$.

for every B and $C \in \mathcal{B}(F)$.

Example

The canonical mapping $\varphi : \mathcal{B}(F) \rightarrow \mathcal{P}(Q \times Q)$ induced by an automaton \mathcal{A} .

Towards quasi-recognizability

Definition (Adequate premorphism)

A mapping $\varphi : \mathcal{B}(F) \rightarrow S$ with S quasi-inverse such that:

- **unit:** $\varphi(1) = 1$,
- **monotonic:** if $B \leq C$ then $\varphi(B) \leq \varphi(C)$,
- **disjoint product:** if $\exists B * C$ then $\varphi(B * C) = \varphi(B) \cdot \varphi(C)$,
- **projections:** $\varphi(B^L) = (\varphi(B))^L$ and $\varphi(B^R) = (\varphi(B))^R$.

for every B and $C \in \mathcal{B}(F)$.

Example

The canonical mapping $\varphi : \mathcal{B}(F) \rightarrow \mathcal{P}(Q \times Q)$ induced by an automaton \mathcal{A} .

Towards quasi-recognizability

Definition (Adequate premorphism)

A mapping $\varphi : \mathcal{B}(F) \rightarrow S$ with S quasi-inverse such that:

- **unit:** $\varphi(1) = 1$,
- **monotonic:** if $B \leq C$ then $\varphi(B) \leq \varphi(C)$,
- **disjoint product:** if $\exists B * C$ then $\varphi(B * C) = \varphi(B) \cdot \varphi(C)$,
- **projections:** $\varphi(B^L) = (\varphi(B))^L$ and $\varphi(B^R) = (\varphi(B))^R$.

for every B and $C \in \mathcal{B}(F)$.

Example

The canonical mapping $\varphi : \mathcal{B}(F) \rightarrow \mathcal{P}(Q \times Q)$ induced by an automaton \mathcal{A} .

Towards quasi-recognizability

Definition (Adequate premorphism)

A mapping $\varphi : \mathcal{B}(F) \rightarrow S$ with S quasi-inverse such that:

- **unit:** $\varphi(1) = 1$,
- **monotonic:** if $B \leq C$ then $\varphi(B) \leq \varphi(C)$,
- **disjoint product:** if $\exists B * C$ then $\varphi(B * C) = \varphi(B) \cdot \varphi(C)$,
- **projections:** $\varphi(B^L) = (\varphi(B))^L$ and $\varphi(B^R) = (\varphi(B))^R$.

for every B and $C \in \mathcal{B}(F)$.

Example

The canonical mapping $\varphi : \mathcal{B}(F) \rightarrow \mathcal{P}(Q \times Q)$ induced by an automaton \mathcal{A} .

Towards quasi-recognizability

Definition (Adequate premorphism)

A mapping $\varphi : \mathcal{B}(F) \rightarrow S$ with S quasi-inverse such that:

- **unit:** $\varphi(1) = 1$,
- **monotonic:** if $B \leq C$ then $\varphi(B) \leq \varphi(C)$,
- **disjoint product:** if $\exists B * C$ then $\varphi(B * C) = \varphi(B) \cdot \varphi(C)$,
- **projections:** $\varphi(B^L) = (\varphi(B))^L$ and $\varphi(B^R) = (\varphi(B))^R$.

for every B and $C \in \mathcal{B}(F)$.

Example

The canonical mapping $\varphi : \mathcal{B}(F) \rightarrow \mathcal{P}(Q \times Q)$ induced by an automaton \mathcal{A} .

Towards quasi-recognizability

Definition (Adequate premorphism)

A mapping $\varphi : \mathcal{B}(F) \rightarrow S$ with S quasi-inverse such that:

- **unit:** $\varphi(1) = 1$,
- **monotonic:** if $B \leq C$ then $\varphi(B) \leq \varphi(C)$,
- **disjoint product:** if $\exists B * C$ then $\varphi(B * C) = \varphi(B) \cdot \varphi(C)$,
- **projections:** $\varphi(B^L) = (\varphi(B))^L$ and $\varphi(B^R) = (\varphi(B))^R$.

for every B and $C \in \mathcal{B}(F)$.

Example

The canonical mapping $\varphi : \mathcal{B}(F) \rightarrow \mathcal{P}(Q \times Q)$ induced by an automaton \mathcal{A} .

Quasi-recognizable languages

Definition

A language $L \subseteq \mathcal{B}(F)$ is **quasi-recognizable** (QREC) when there exists a finite QI-monoid S and an adequate premorphism $\varphi : \mathcal{B}(F) \rightarrow S$ such that $L = \varphi^{-1}(\varphi(L))$.

Lemma (Effectiveness)

For every $B \in \mathcal{B}(F)$, the image $\varphi(B)$ of B is computable in linear time in B .

Theorem (See [Jan13c, Jan13a])

$$\text{QREC} = \text{Bool}(\text{MSO}^\uparrow)$$

Quasi-recognizable languages

Definition

A language $L \subseteq \mathcal{B}(F)$ is **quasi-recognizable** (QREC) when there exists a finite QI-monoid S and an adequate premorphism $\varphi : \mathcal{B}(F) \rightarrow S$ such that $L = \varphi^{-1}(\varphi(L))$.

Lemma (Effectiveness)

For every $B \in \mathcal{B}(F)$, the image $\varphi(B)$ of B is computable in linear time in B .

Theorem (See [Jan13c, Jan13a])

$$\text{QREC} = \text{Bool}(\text{MSO}^\uparrow)$$

BOOL(MSO[↑]) vs MSO ?

Theorem

BOOL(MSO[↑]) is strictly included into MSO

Proof.

Let $L = \{a^{2n}a^{-2n} : n \in \mathbb{N}\}$. Then L is definable in MSO while L cannot be recognized by a monotonic function in a finite set. \square

Remark

This implies that QREC is not closed under product nor star. Need to restrict to “positive” birooted trees [DJ14].

BOOL(MSO[↑]) vs MSO ?

Theorem

BOOL(MSO[↑]) is strictly included into MSO

Proof.

Let $L = \{a^{2n}a^{-2n} : n \in \mathbb{N}\}$. Then L is definable in MSO while L cannot be recognized by a monotonic function in a finite set. \square

Remark

This implies that QREC is not closed under product nor star. Need to restrict to “positive” birooted trees [DJ14].

BOOL(MSO[↑]) vs MSO ?

Theorem

BOOL(MSO[↑]) is strictly included into MSO

Proof.

Let $L = \{a^{2n}a^{-2n} : n \in \mathbb{N}\}$. Then L is definable in MSO while L cannot be recognized by a monotonic function in a finite set. \square

Remark

This implies that QREC is not closed under product nor star. Need to restrict to “positive” birooted trees [DJ14].

The partial algebra approach (last results)

From partial algebra theory (see e.g., [Bur86]).

Definition (*-congruence)

An equivalence \simeq of birooted F -tree is a closed *-congruence when:

- if $B \simeq C$ then $B^L \simeq C^L$ and $B^R \simeq C^R$,
- if $B \simeq C$ and $B' \simeq C'$ then $\exists B * B' = \exists C * C'$,

for every $B, B', C, C' \in \mathcal{B}(F)$.

Theorem (Syntactic congruence)

*For every $X \subseteq \mathcal{B}(F)$ there exists a greatest closed *-congruence \simeq_X such that, for every $B, C \in \mathcal{B}(F)$, if $B \simeq_X C$ then $B \in X \Leftrightarrow C \in X$.*

The partial algebra approach (last results)

From partial algebra theory (see e.g., [Bur86]).

Definition (*-congruence)

An equivalence \simeq of birooted F -tree is a closed *-congruence when:

- if $B \simeq C$ then $B^L \simeq C^L$ and $B^R \simeq C^R$,
- if $B \simeq C$ and $B' \simeq C'$ then $\exists B * B' = \exists C * C'$,

for every $B, B', C, C' \in \mathcal{B}(F)$.

Theorem (Syntactic congruence)

*For every $X \subseteq \mathcal{B}(F)$ there exists a greatest closed *-congruence \simeq_X such that, for every $B, C \in \mathcal{B}(F)$, if $B \simeq_X C$ then $B \in X \Leftrightarrow C \in X$.*

The partial algebra approach (last results)

Theorem

A language $X \subseteq \mathcal{B}(F)$ is MSO if and only if \simeq_X is of finite index.

Theorem

A language $X \subseteq \mathcal{B}(F)$ is QREC if and only if \simeq_X is of finite index and the size of chain $B_1 \leq B_1 \leq \dots \leq B_{n+1}$ with $B_i \not\leq_X B_{i+1}$ is uniformly bounded.

Remark

There is a strong incentive to consider strongly adequate premorphism, that is, adequate premorphism such that, moreover:

- if $\varphi(B) \leq \varphi(C)$ then there exists B' such that $\varphi(B') = \varphi(B)$ and $B' \leq C$.

The partial algebra approach (last results)

Theorem

A language $X \subseteq \mathcal{B}(F)$ is MSO if and only if \simeq_X is of finite index.

Theorem

A language $X \subseteq \mathcal{B}(F)$ is QREC if and only if \simeq_X is of finite index and the size of chain $B_1 \leq B_1 \leq \dots \leq B_{n+1}$ with $B_i \not\leq_X B_{i+1}$ is uniformly bounded.

Remark

There is a strong incentive to consider strongly adequate premorphism, that is, adequate premorphism such that, moreover:

- if $\varphi(B) \leq \varphi(C)$ then there exists B' such that $\varphi(B') = \varphi(B)$ and $B' \leq C$.

The partial algebra approach (last results)

Theorem

A language $X \subseteq \mathcal{B}(F)$ is MSO if and only if \simeq_X is of finite index.

Theorem

A language $X \subseteq \mathcal{B}(F)$ is QREC if and only if \simeq_X is of finite index and the size of chain $B_1 \leq B_1 \leq \dots \leq B_{n+1}$ with $B_i \not\leq_X B_{i+1}$ is uniformly bounded.

Remark

There is a strong incentive to consider strongly adequate premorphism, that is, adequate premorphism such that, moreover:

- if $\varphi(B) \leq \varphi(C)$ then there exists B' such that $\varphi(B') = \varphi(B)$ and $B' \leq C$.

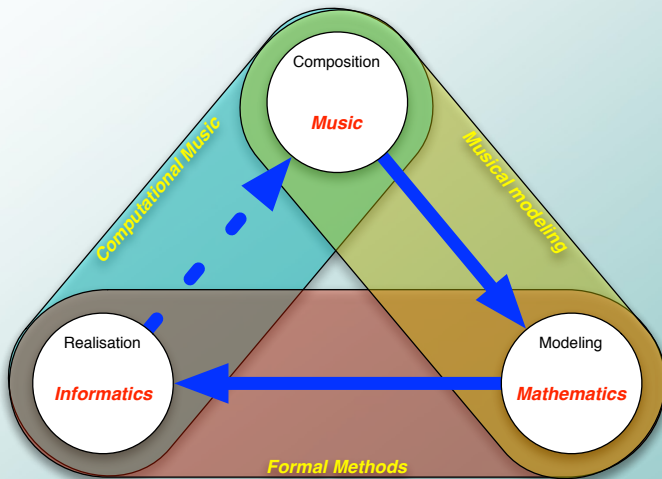
6. Conclusion

Back to music, plus a bit of philosophy

SimpleTuilesLooper (by F. Berthaut)

Only if time permits...

The virtuous circle of research in CS



[BJM12] F. Berthaut, D. Janin, and B. Martin.

Advanced synchronization of audio or symbolic musical patterns: an algebraic approach.

International Journal of Semantic Computing,
6(4):409–427, 2012.

[Boj08] M. Bojańczyk.

Tree-walking automata.

In *2nd Int. Conf. on Language and Automata Theory and Applications (LATA)*, volume 5196 of *LNCS*. Springer, 2008.

[Bur86] P. Burmeister.

A Model Theoretic Oriented Approach to Partial Algebras.

Akademie-Verlag, 1986.

[DH88] P. Desain and H. Honing.

LOCO: a composition microworld in Logo.

Computer Music Journal, 12(3):30–42, 1988.

- [DJ13] A. Dicky and D. Janin.
Two-way automata and regular languages of overlapping tiles.
Research report RR-1463-12, LaBRI, Université de Bordeaux, 2013.
- [DJ14] E. Dubourg and D. Janin.
Algebraic tools for the overlapping tile product.
In *Language and Automata Theory and Applications (LATA)*, Madrid, Spain, 2014. Springer.
- [Fou77] J. Fountain.
Right PP monoids with central idempotents.
Semigroup Forum, 13:229–237, 1977.
- [Jan12a] D. Janin.
Quasi-inverse monoids (and premorphisms).
Research report RR-1459-12 (revised 11/2013), LaBRI, Université de Bordeaux, 2012.
- [Jan12b] D. Janin.

Vers une modélisation combinatoire des structures rythmiques simples de la musique.

Revue Francophone d'Informatique Musicale (RFIM), 2, 2012.

[Jan13a] D. Janin.

Algebras, automata and logic for languages of labeled birooted trees.

In *Int. Col. on Aut., Lang. and Programming (ICALP)*, volume 7966 of *LNCS*, pages 318–329, Riga, Latvia, 2013. Springer.

[Jan13b] D. Janin.

On languages of one-dimensional overlapping tiles.

In *Int. Conf. on Current Trends in Theo. and Prac. of Comp. Science (SOFSEM)*, volume 7741 of *LNCS*, pages 244–256, Spindlerův Mlýn, Czech Republic, 2013. Springer.

[Jan13c] D. Janin.

Overlapping tile automata.

In *8th International Computer Science Symposium in Russia (CSR)*, volume 7913 of *LNCS*, pages 431–443, Ekaterinburg, Russia, 2013. Springer.

[Jan13d] D. Janin.

Walking automata in the free inverse monoid.

Research report RR-1464-12, LaBRI, Université de Bordeaux, 2013.

(revised May 2013).

[Jan14] D. Janin.

Towards a higher dimensional string theory for the modeling of computerized systems, volume 8327 of *LNCS*, pages 7–20.

Springer, Novy Smokovec, Slovaquia, 2014.

[JBD13] D. Janin, F. Berthaut, and M. DeSainteCatherine.

Multi-scale design of interactive music systems : the libTuiles experiment.

In *10th Conference on Sound and Music Computing (SMC)*, Stockholm, Sweden, 2013.

- [Law91] M. V. Lawson.
Semigroups and ordered categories. I. the reduced case.
Journal of Algebra, 141(2):422 – 462, 1991.
- [Law98] M. V. Lawson.
Inverse Semigroups : The theory of partial symmetries.
World Scientific, 1998.
- [Lee87] J. Leech.
Constructing inverse monoids from small categories.
Semigroup Forum, 36:89–116, 1987.
- [Ste90] J.B. Stephen.
Presentations of inverse monoids.
Journal of Pure and Applied Algebra, 63:81–112, 1990.
- [Tho97] W. Thomas.
Chap. 7. Languages, automata, and logic.

In *Handbook of Formal Languages, Vol. III*, pages 389–455. Springer-Verlag, Berlin Heidelberg, 1997.

Thanks for your attention !