# reflections on a geometry of processes

clemens grabmayer
jan willem klop
bas luttik

Some memories

Some questions



Bertinoro August 2005

*May 1975, cherry orchard in the Betuwe*

Jan Bergstra and Jan Willem Klop started working on process algebra after a lecture by Jaco in Utrecht in June 1982. They tackled the open problem he posed of solving unguarded recursion equations in the topological model of De Bakker and Zucker [1982]. Their solution was this: in the case of a finite set of atomic actions, they created the axiomatic system *Process Algebra* PA for processes. The theory PA had an initial algebra $A_\omega$ and a system of projections $A_n$ that modelled the execution of processes for $n$ steps, for $n = 1, 2, \ldots$ . These projections were also models of PA and the algebras formed an inverse sequence with inverse or projective limit $A_\infty$, which was again a model of PA. They proved that all recursion equations have solutions in all the $A_n$ and so in the $A_\infty$. Since the $A_\infty$ can be embedded in the De Bakker-Zucker model of processes, the problem was solved.

Problem: Expansion Theorem

LFCS · People · Publications

## Laboratory for Foundations of Computer Science

LFCS Theory Seminar
Room 2511, JCMB, King's Buildings
4pm, Tuesday 13th January 1998

Title: Unique Fixed Points for Unguarded Recursion

Speaker: Rob van Glabbeek (Stanford University, USA)

Problem: we did not know SOS rules

To facilitate computations with processes (both BT and LT) we devise a

## PROCESS ALGEBRA.

PA1. $x + y = y + x$

PA2. $x + (y + z) = (x + y) + z$

PA3. $x + x = x$

PA4. $(x + y)z = xz + yz$

$z(x+y) = zx + zy$ (For LT)

PA5. $x(yz) = (xy)z$

PA6. $(x+y) ⫫ z = x ⫫ z + y ⫫ z$

PA7. $ax ⫫ y = a(x ⫫ y + y ⫫ x)$

PA8. $a ⫫ y = ay$

DEFINITION: $x \parallel y = x ⫫ y + y ⫫ x$.

Examples of process algebras are

$\mathbb{P}$ ( ⫫ is easy to define on $\mathbb{P}$ )

$\mathbb{C}$ ( " ) $\models z(x+y) = zx + zy$

$\mathbb{A}$ the term algebra (or initial algebra) determined by PA 1,...,8.

⋮

Jos: precursor Bekic; Hennessy

We write $x^{n} = x\,x \ldots x$ (n times)

$$x^{\underline{\underline{n}}} = x /\!/ x /\!/ x \ldots /\!/ x \quad \text{(n times)}$$

We have $x /\!/ y = x \llcorner\!\llcorner\, y + y \llcorner\!\llcorner\, x = y \llcorner\!\llcorner\, x + x \llcorner\!\llcorner\, y = y /\!/ x$

$x /\!/ (y /\!/ z) = (x /\!/ y) /\!/ z$ (induction to the structure of elements $\in \mathbb{A}$)

$(x \llcorner\!\llcorner\, y) \llcorner\!\llcorner\, z = x \llcorner\!\llcorner\, (y /\!/ z)$ (idem)

$$\begin{aligned} x /\!/ y /\!/ z = \; & x \llcorner\!\llcorner\, (y /\!/ z) \; + \\ & y \llcorner\!\llcorner\, (x /\!/ z) \; + \\ & z \llcorner\!\llcorner\, (x /\!/ y) . \end{aligned}$$

in particular:

$$x /\!/ x /\!/ x = x^{\underline{\underline{3}}} = x \llcorner\!\llcorner\, (x /\!/ x) = x \llcorner\!\llcorner\, x^{\underline{\underline{2}}} .$$

In general:

$$\boxed{x^{\underline{\underline{n+1}}} = x \llcorner\!\llcorner\, x^{\underline{\underline{n}}}}$$

Now we have

$$((x)_n)_m = (x)_{min(n,m)}$$

$$(x+y)_n = ((x)_n + (y)_n)_n$$

$$(x y)_n = ((x)_n\,(y)_{n-1})_n$$

$$(x \mathbin{\lfloor\!\lfloor} y)_n = ((x)_n \mathbin{\lfloor\!\lfloor} (y)_{n-1})_n$$

$$(x y)_1 = (x)_1$$

$$(x \mathbin{\lfloor\!\lfloor} y)_1 = (x)_1 .$$

(Note the similarity between $\circ$ and $\lfloor\!\lfloor$.)

Jan: aha, this solves our problem

# THE QUESTION.

In the course of assigning a semantics (?) to $\mu$- statements one has to prove the convergence of certain sequences of elements in $\mathbb{A}$.

These sequences have the form

$$\underline{q}\,,\ s(\underline{q})\,,\ s(s(\underline{q}))\,,\ s(s(s(\underline{q})))\,,\ \ldots$$

where $\underline{q} \in \mathbb{A}$ and $s(x)$ is an expression built from $x$, $a, b, c, \ldots$, $+$, $\cdot$ and $\|$.

We will call such sequences

iteration sequences (generated by $s(x)$, starting with $\underline{q}$)

'Convergence' refers to the

metric $d$

such that

$$d(p,\underline{q}) = \begin{cases} 2^{-\min\{n \,/\, (p)_n \neq (\underline{q})_n\}} & \text{if } \exists n.. \\ 0 \quad \text{else} & (\text{i.e. if } p = \underline{q}) \end{cases}$$

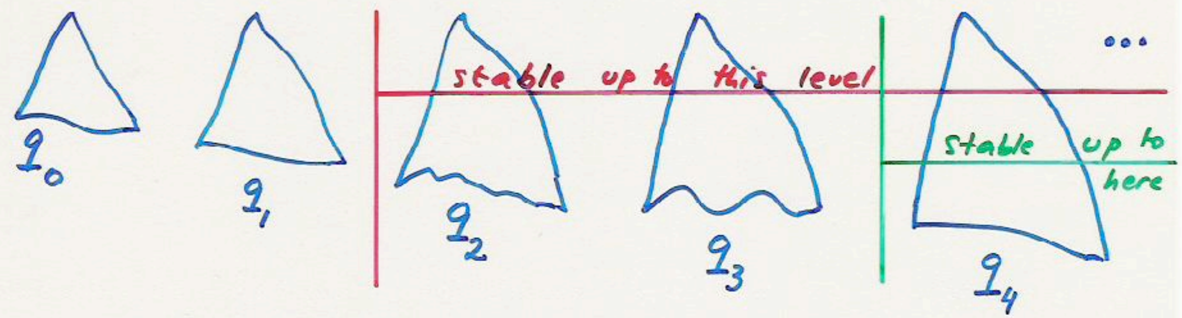
So $d(p,\underline{q}) = \dfrac{1}{2^{\ell}}$

where $\ell$ is such that $p = \underline{q}$ mod. $\ell-1$ but $p \neq \underline{q}$ mod. $\ell$.

What does it mean for a (general) sequence

$q_0 , q_1 , q_2 , \cdots$

to be convergent in this sense?




$$\boxed{\begin{array}{l}\text{Convergence} \quad = \\ \text{Stabilization modulo } n, \text{ for every } n.\end{array}}$$

( The sequence $q_0 , q_1 , \cdots$ <u>stabilizes mod. $n$</u> means:
the sequence of approximations

$(q_0)_n , (q_1)_n , (q_2)_n , \cdots\cdots$

will be <u>eventually constant</u>. )

In general, for 'guarded' $s(x)$, no problem.

Example of unguarded $s(x)$:

$$(x \parallel x) + ab.$$

## THE SOLUTION.

First we prove that for every $q \in A$ the sequence

$$\underline{q}, \quad \underline{q} \parallel \underline{q}, \quad \underline{q} \parallel \underline{q} \parallel \underline{q}, \ldots, \quad \underline{q}^{\underline{k}}, \quad \ldots$$

stabilizes modulo $n$ $(\forall n \geq 1)$.

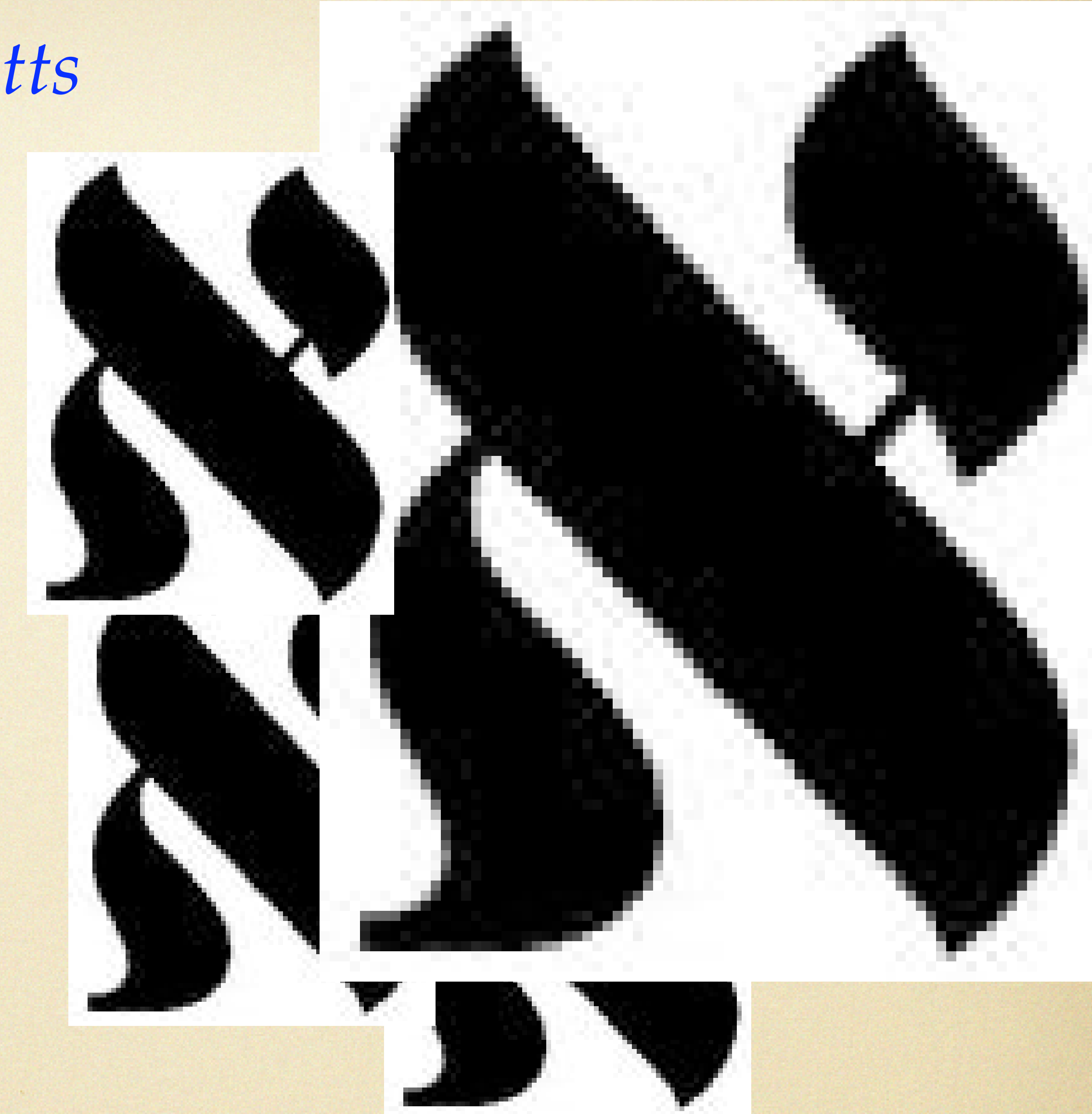
We will now state and prove the main theorem of this paper, saying that every sequence q, s(q), $s^2$(q), ... must eventually be constant modulo n.

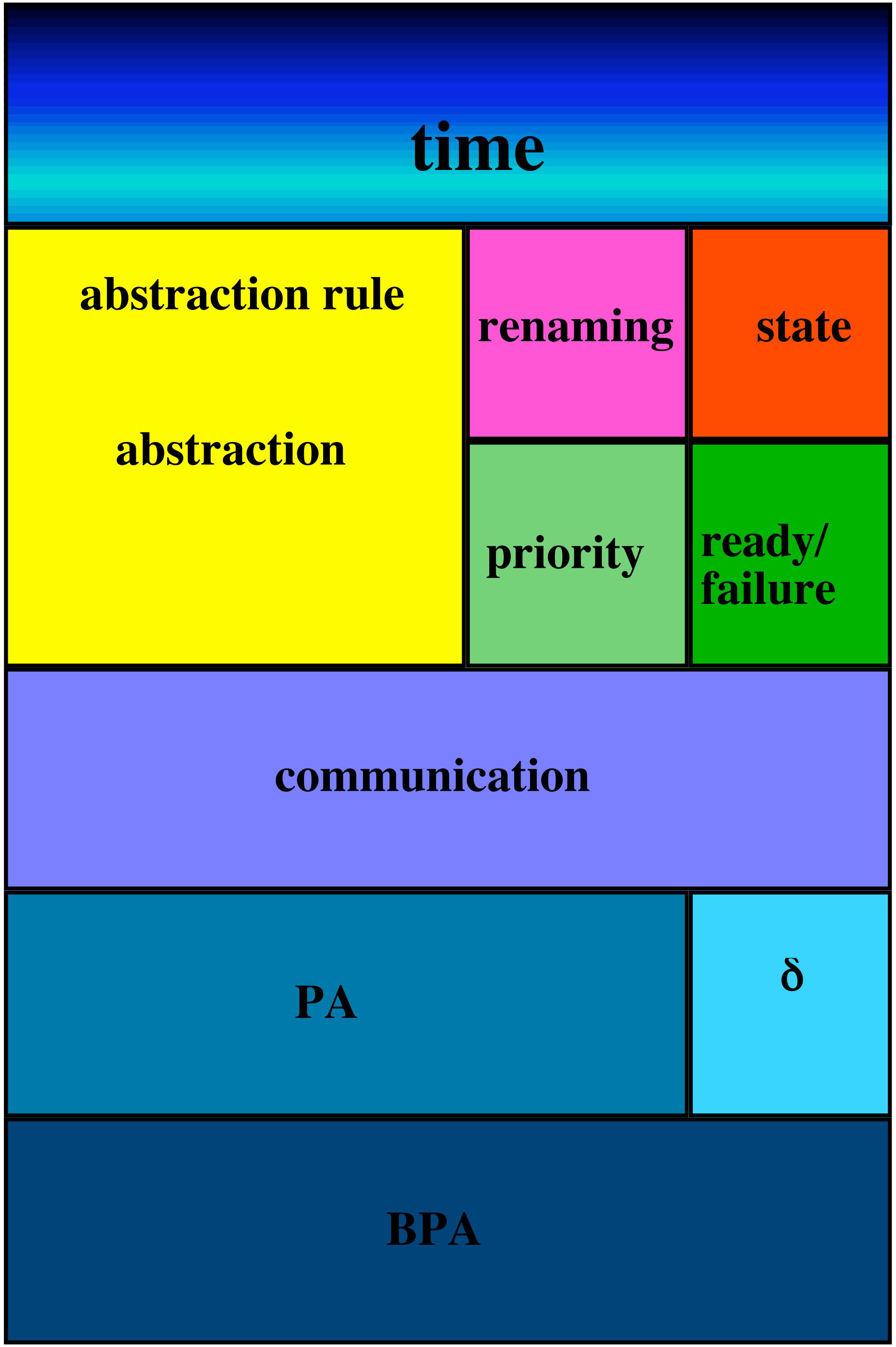
For *guarded* expressions like e.g. $s(X) = aX + b(cX \parallel X^3) + d$ this is clear since iterating s(X) yields a tree which develops itself in such a way that an increasing part of it is fixed.

But even for simple terms as $s(X) = (X \parallel X) + ab$ the situation is at first sight not at all clear: in each step of the iteration the whole tree including the top is again in 'motion'.

THEOREM. *Let* $q \in A^\omega$ *and let* $s(X) \in \mathrm{EXP}$ *have only* X *as free variable. Then the iteration sequence* q, s(q), s(s(q)), ..., $s^k$(q), ... *stabilizes modulo* n, *for every* $n \geq 1$.

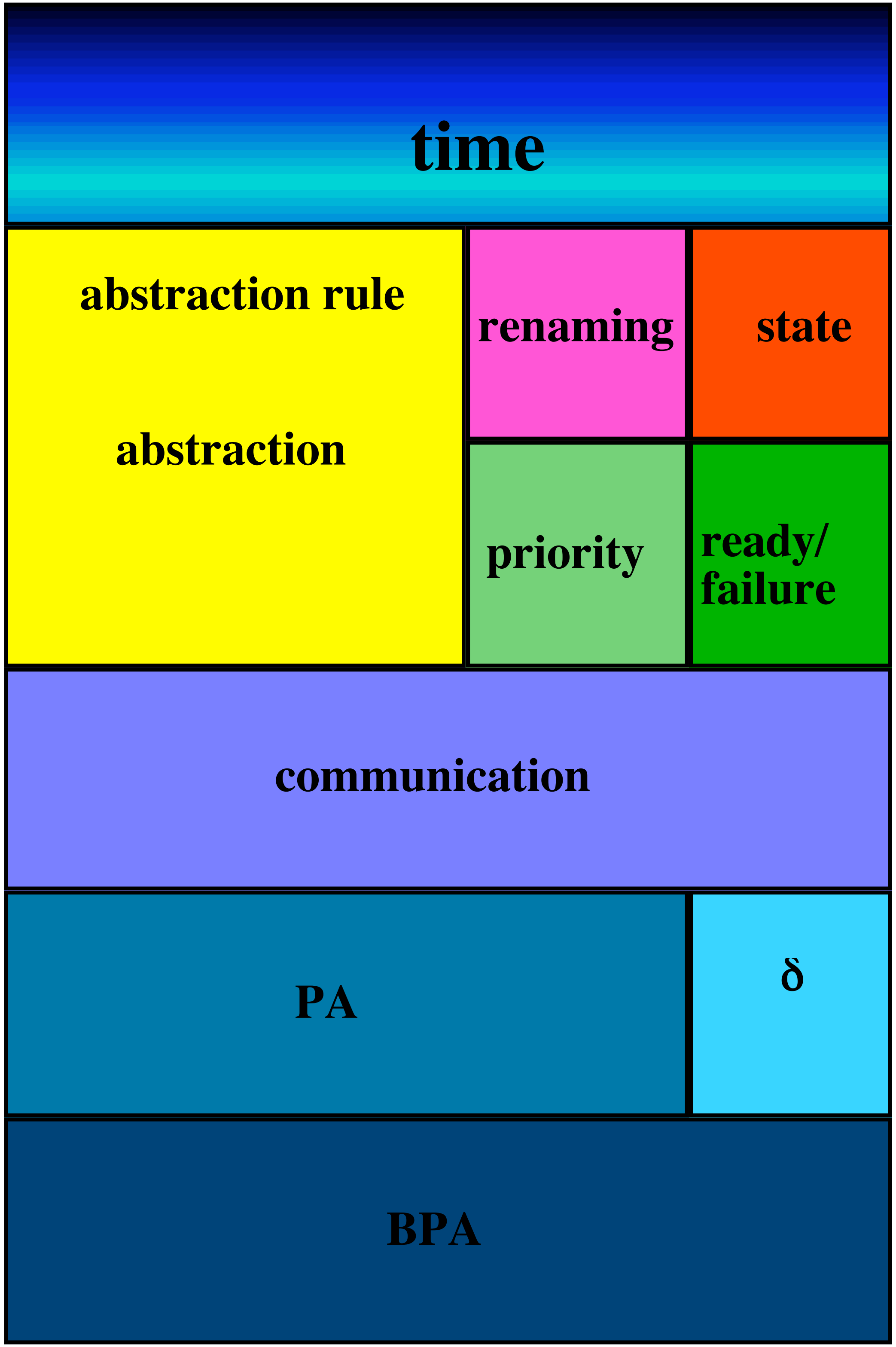
1983, Massachusetts






*Jan: main architect and prime mover*

*Jos, JW: contractors and interior decorators*

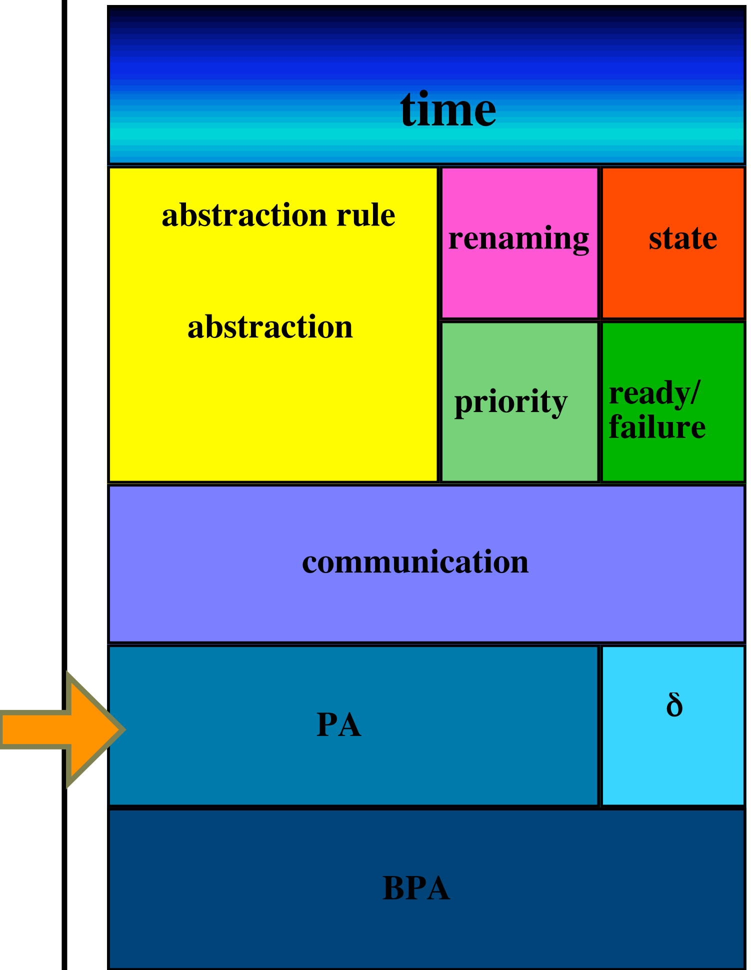



Jan, Jos, Kees

*equational*
*sos*
*branching*

1984 Rob, Frits

Catuscia, Frank, Gert, Karst, Hans, Vagelis, Yurek, John, Alban, Piet, Inge, Jan Friso, Wan, Judy, Jaco, Stefan, Mark, Bas, Simone, Yaroslav, Jun, Natalya, and many many more




*Modular structure of ACP*

$$x + y = y + x$$

$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$(x + y) \cdot z = x.z + y.z$$

$$x.(y.z) = (x.y).z$$

$$x \parallel y = x \mathbin{|\!\!\lfloor} y + y \mathbin{|\!\!\lfloor} x$$

$$a \mathbin{|\!\!\lfloor} x = a.x$$

$$ax \mathbin{|\!\!\lfloor} y = a(x \parallel y)$$

$$(x + y) \mathbin{|\!\!\lfloor} z = x \mathbin{|\!\!\lfloor} z + y \mathbin{|\!\!\lfloor} z$$

Tabel 6.2

*The left merge is an auxiliary operator necessary for a finite axiomatization of merge.*

*PA has unique prime decomposition:*

$$p = p_1 \parallel \ldots \parallel p_n$$

*unique modulo permutation of 'parallel primes'*

*Every process which is recursively defined in* PA *and has an infinite trace, has an eventually periodic trace.*

# Thue-Morse sequence:

M =1001 0110 01101001 0110100110010110 ...

M =1001 0110 01101001 0110100110010110 ...

M = zip M inv(M)

M can be defined in ACP with renaming, or in ACP with ternary communication. With binary communication?

M cannot be defined in PA, since its one single trace is not eventually periodic.

$$
\begin{aligned}
x+y &= y+x \\
x+(y+z) &= (x+y)+z \\
x+x &= x \\
(x+y)\cdot z &= x\cdot z+y\cdot z \\
(x\cdot y)\cdot z &= x\cdot (y\cdot z)
\end{aligned}
$$

Table 1: BPA (Basic Process Algebra)

*context-free grammar in standard form (Greibach normal form)*

$$S \to aB \mid bA$$

$$A \to a \mid aS \mid bAA$$

$$B \to b \mid bS \mid aBB$$

language equality undecidable

*guarded nonlinear recursion system over BPA*

$$S = aB + bA$$

$$A = a + aS + bAA$$

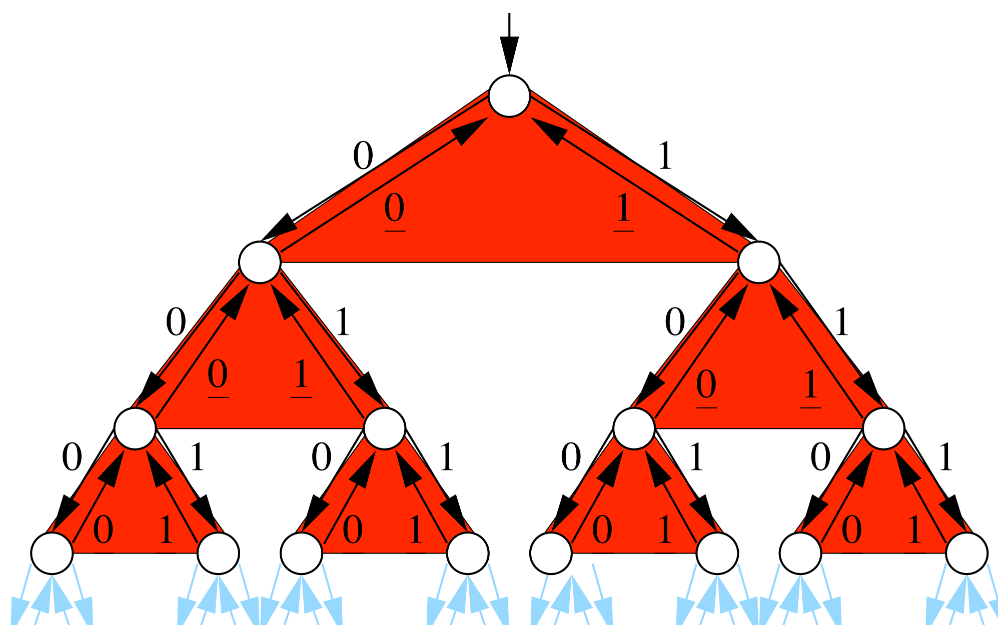
$$B = b + bS + aBB$$

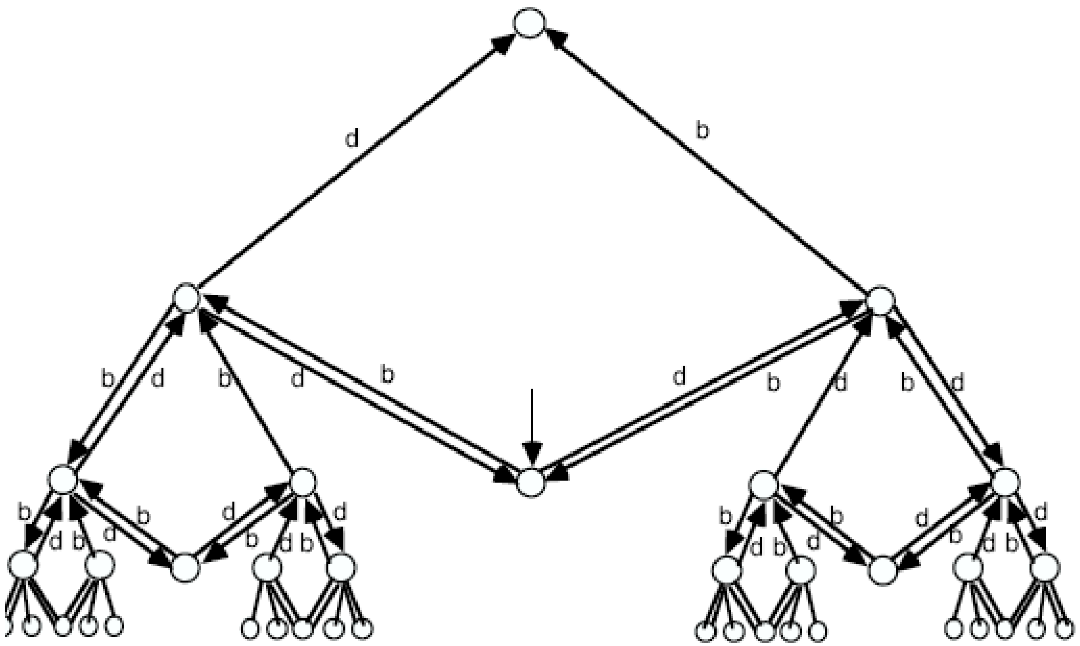
process equality decidable

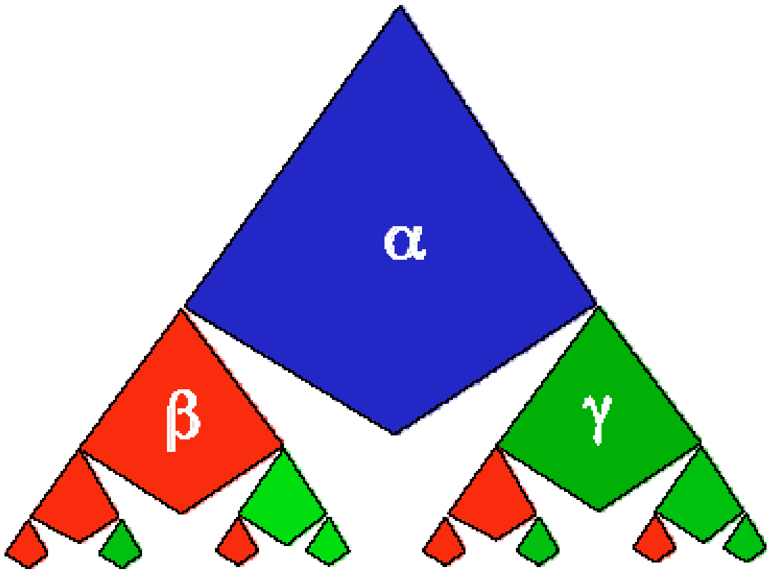
$$
\begin{aligned}
S_\lambda &= 0\cdot S_0 + 1\cdot S_1 \\
S_{d\sigma} &= 0\cdot S_{0d\sigma} + 1\cdot S_{1d\sigma} + \underline{d}\cdot S_\sigma
\end{aligned}
$$
(for $d = 0$ or $d = 1$, and any string $\sigma$)

$$
\begin{aligned}
S &= T\cdot S \\
T &= 0\cdot T_0 + 1\cdot T_1 \\
T_0 &= \underline{0} + T\cdot T_0 \\
T_1 &= \underline{1} + T\cdot T_1
\end{aligned}
$$

Table 2: Stack, an infinite linear and a finite non-linear BPA-specification



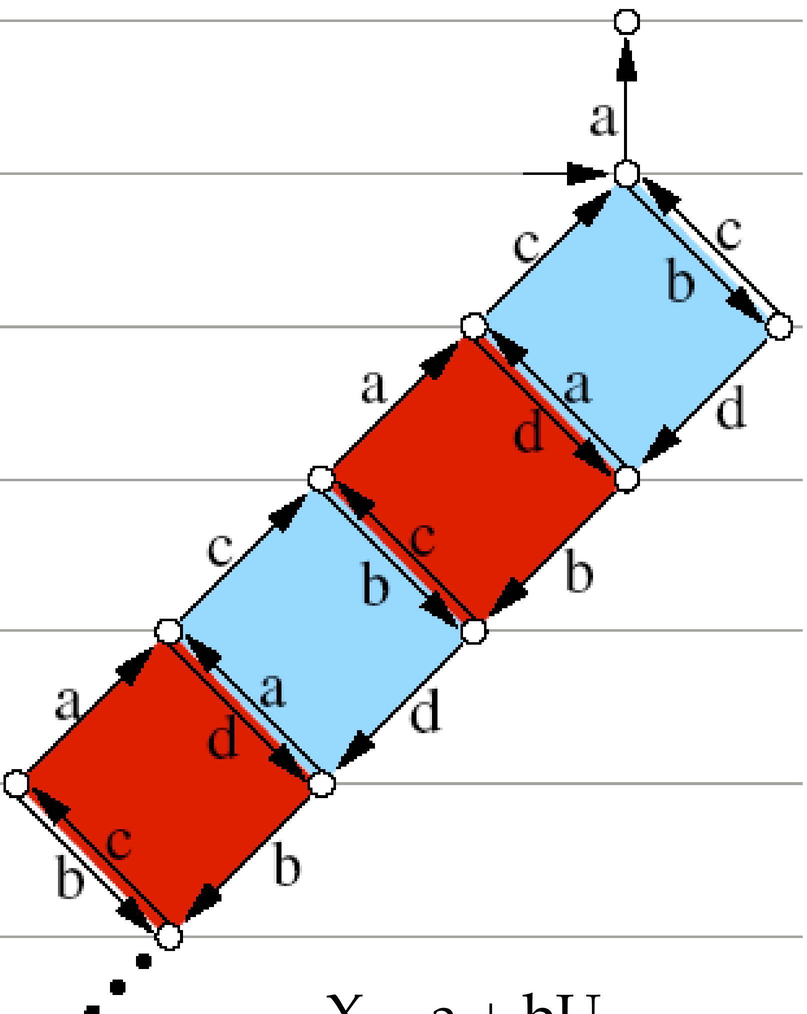





$$X = bY + dZ$$
$$Y = b + bX + dYY$$
$$Z = d + dX + bZZ.$$

*context free language of words*
*with just as many b's as d's*

## type I



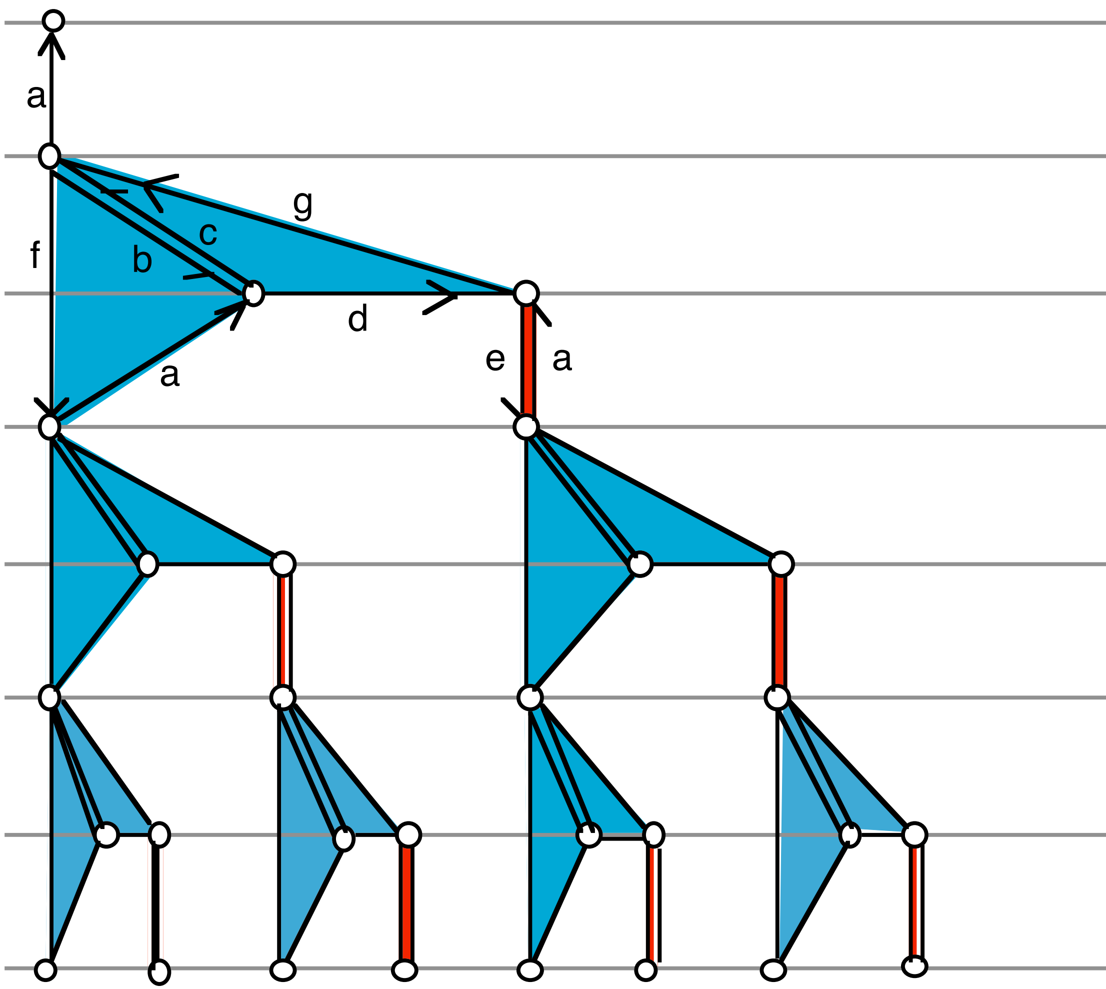

$X = a + bU$

$U = cX + dZX$

$Y = c + dZ$

$Z = aY + bUY$

## type II

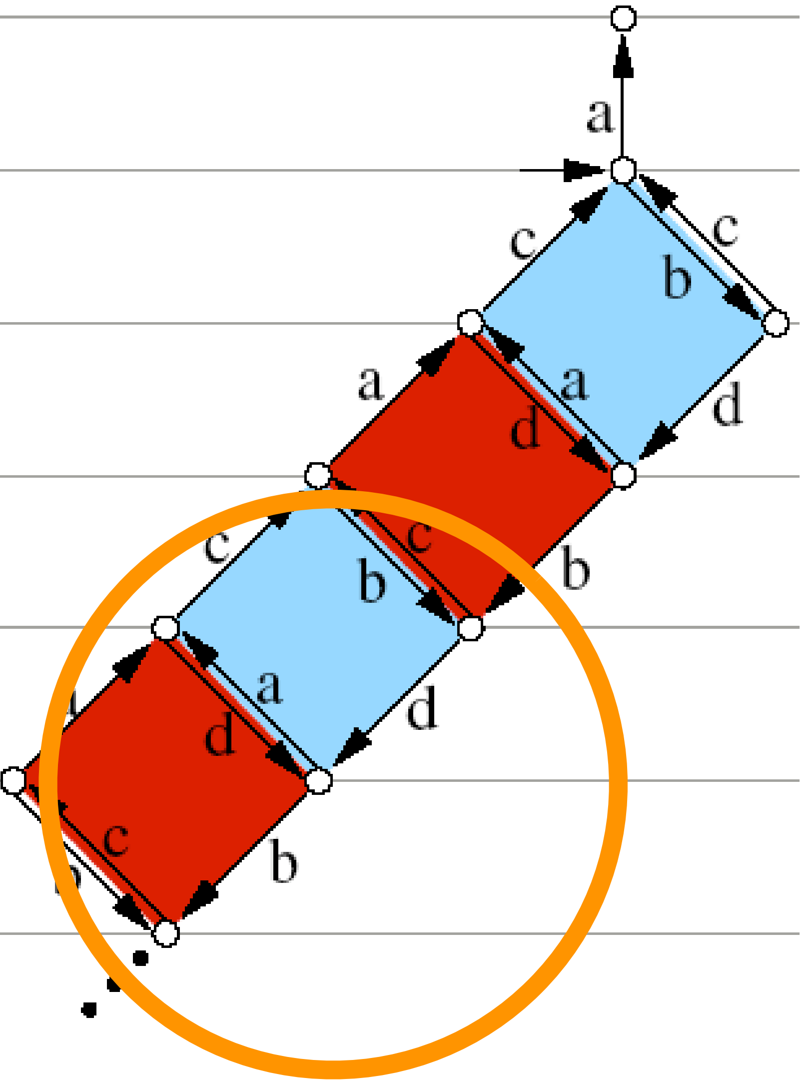



$X = a + bY + fXY$

$Y = cX + dZ$

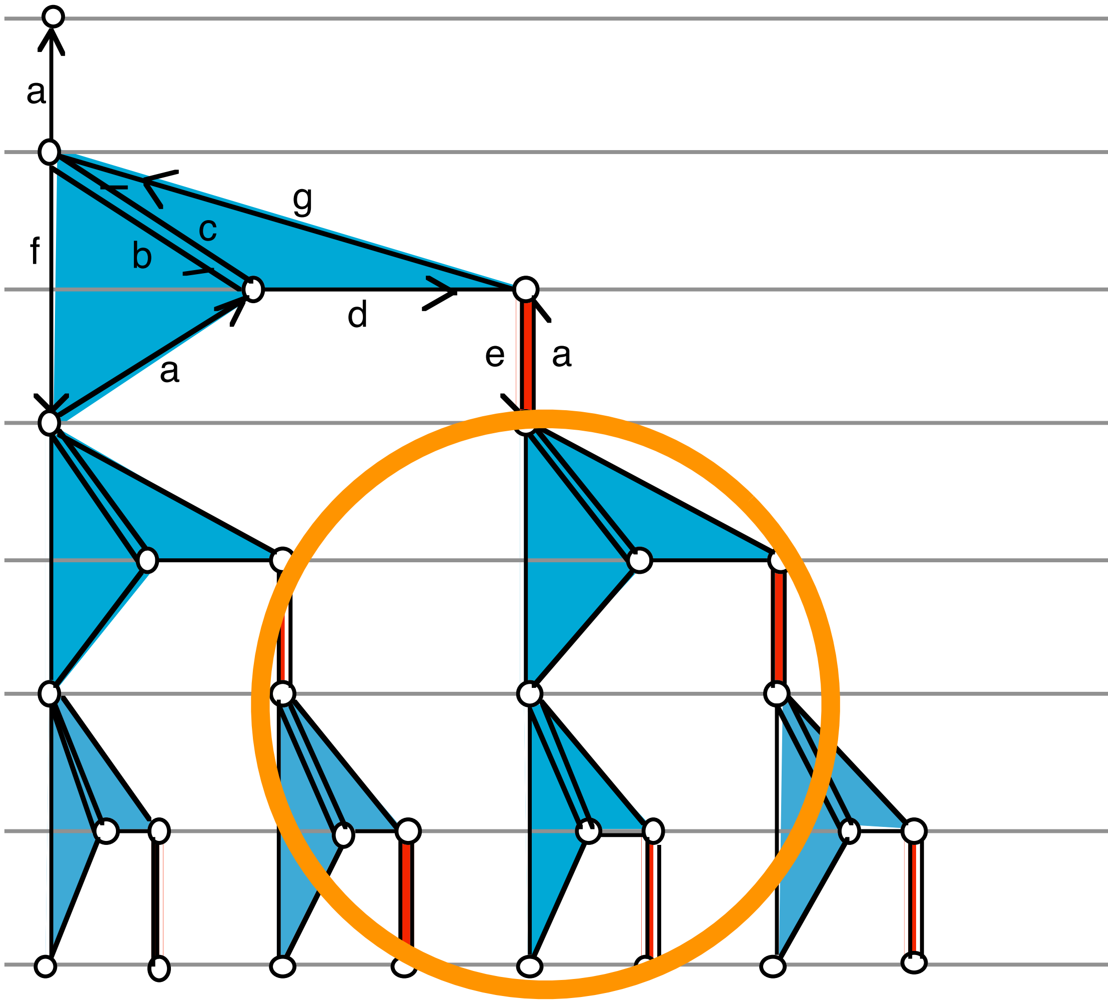
$Z = gX + eXZ.$

*normed graph*

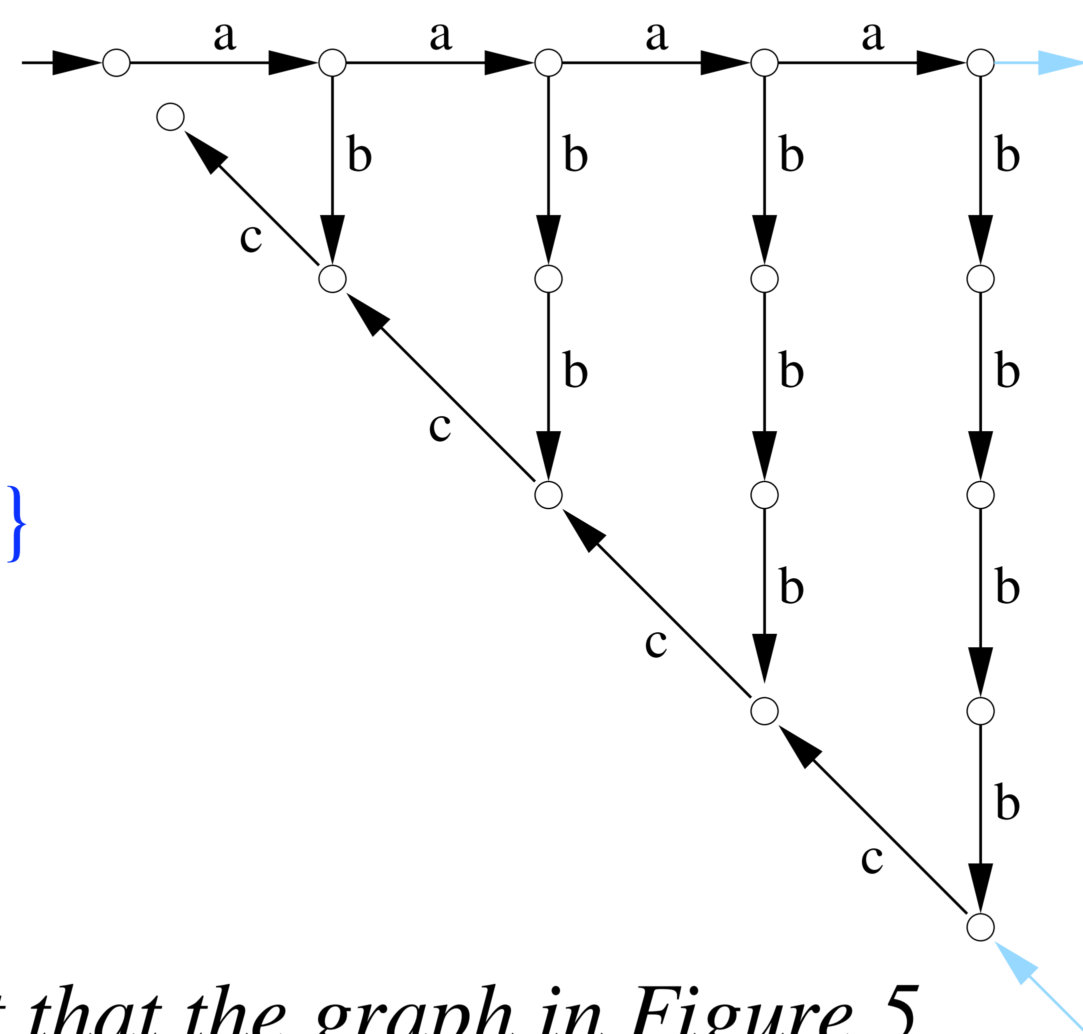



*linear density*

*normed graph*




*exponential*

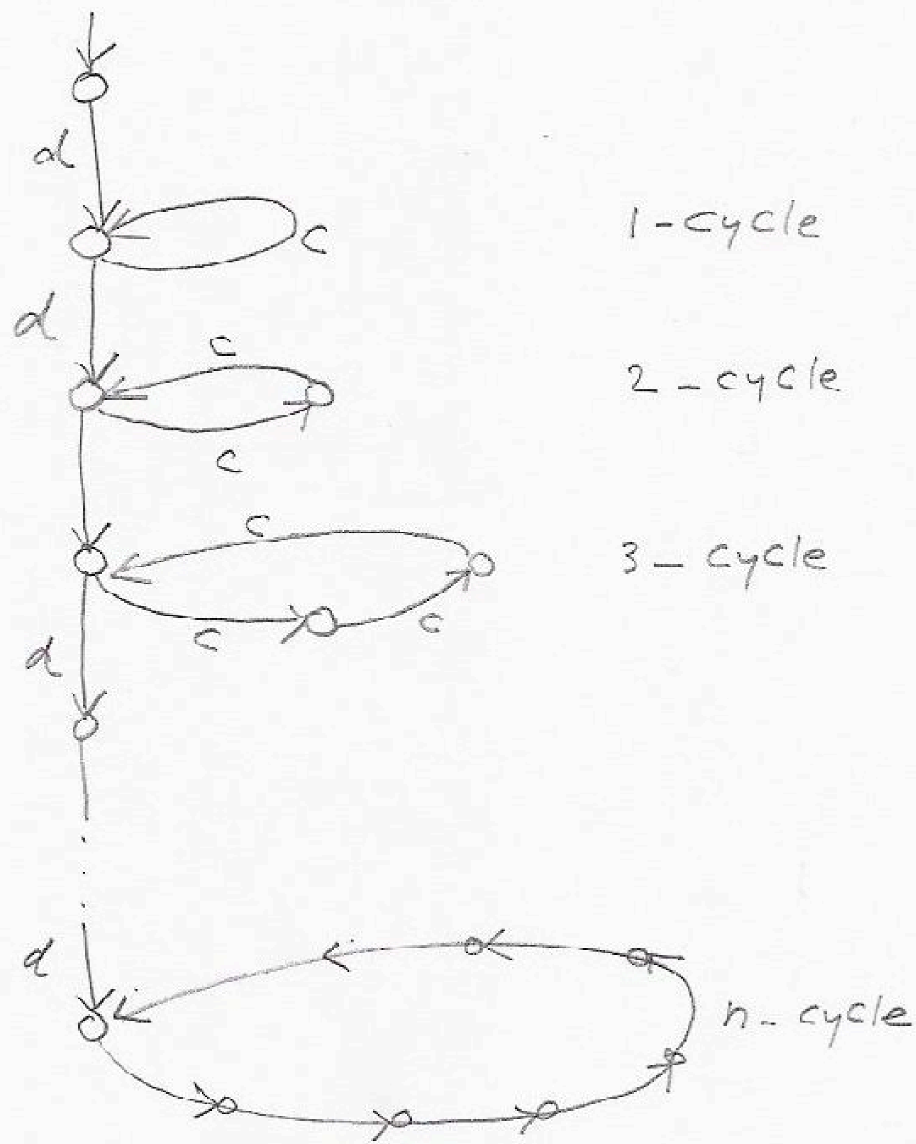



$$L = \{a^n b^n c^n \mid n \geq 1\}$$

**Question 4** *Can the fact that the graph in Figure 5 is not a* BPA*-graph (when established rigorously) be used to conclude that L is not a CFL, applying the correspondence between CFL's and definability in* BPA *as well as the ensuing tree-like periodicity?*

Henk Barendregt:

is this process BPA-definable:



*reopen a cold case: non-BPA definability of BAG*

$$B = a(\underline{a} \parallel B) + b(\underline{b} \parallel B)$$

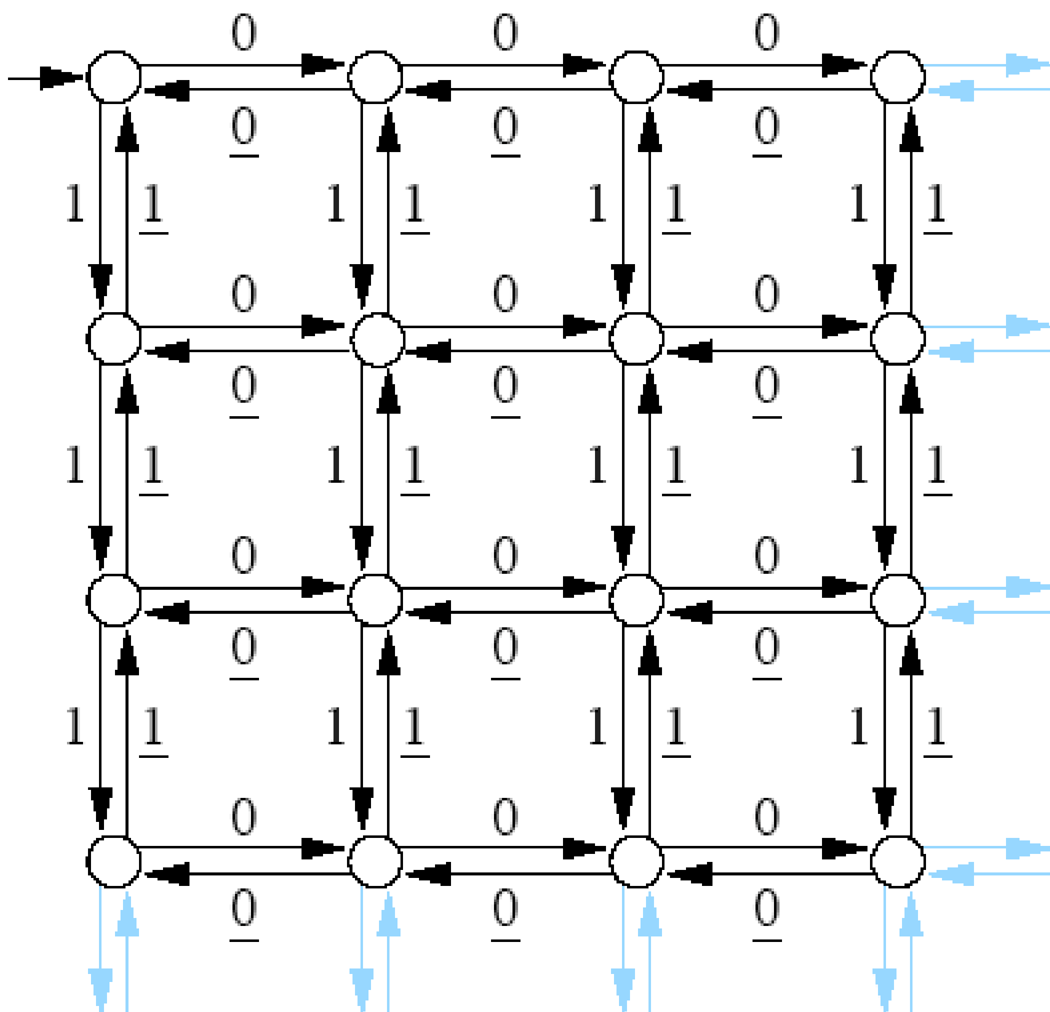
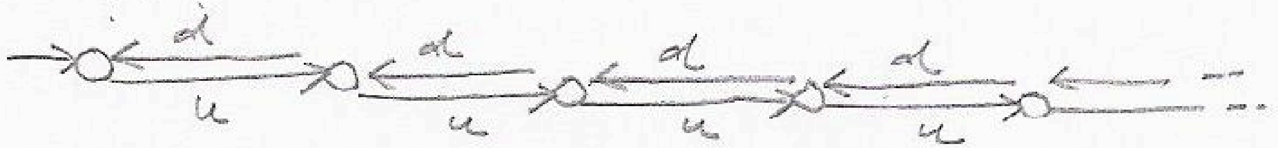


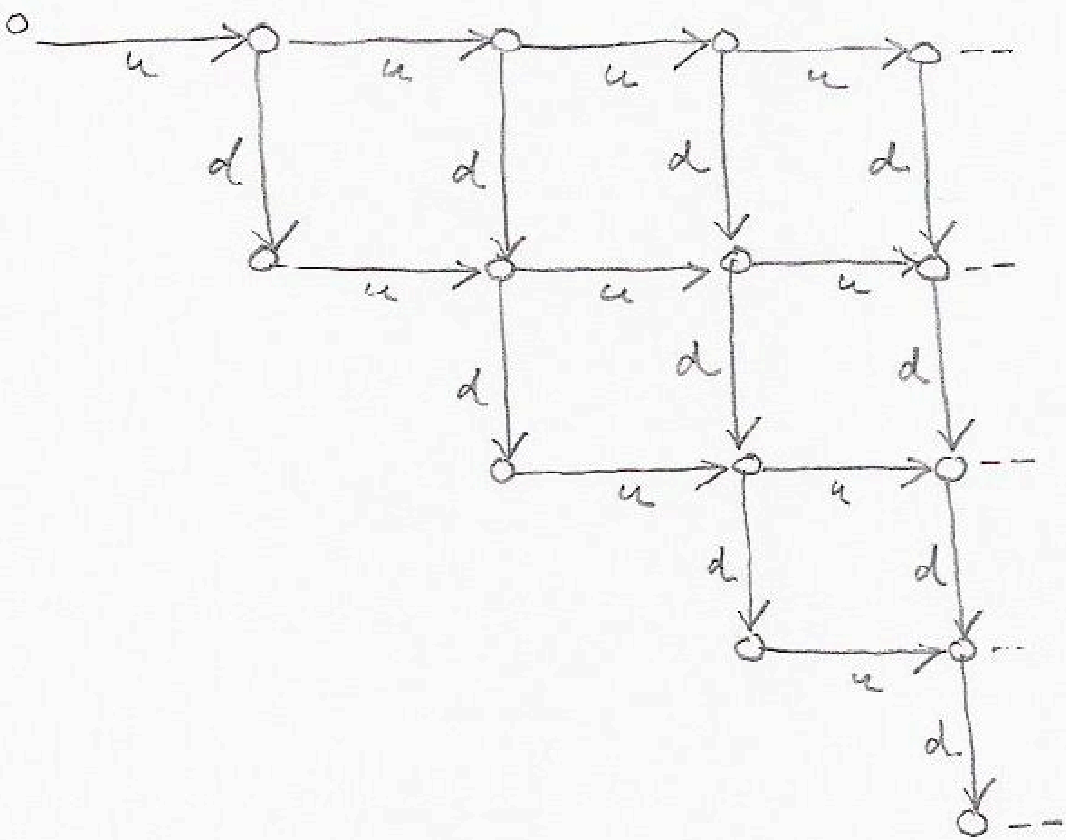

Figure 5: The process Bag.

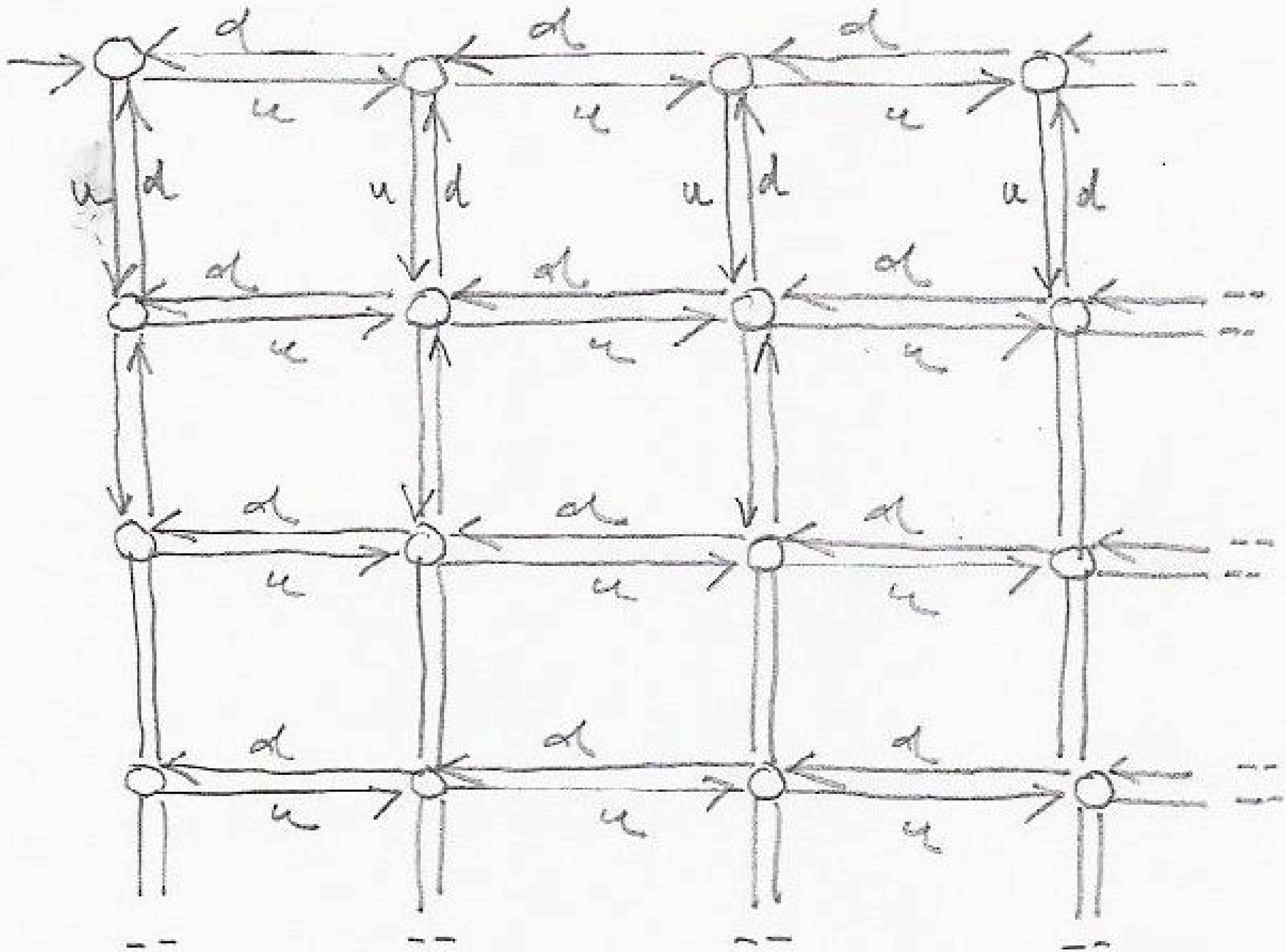
PROBLEM

Counter $C = uDC$

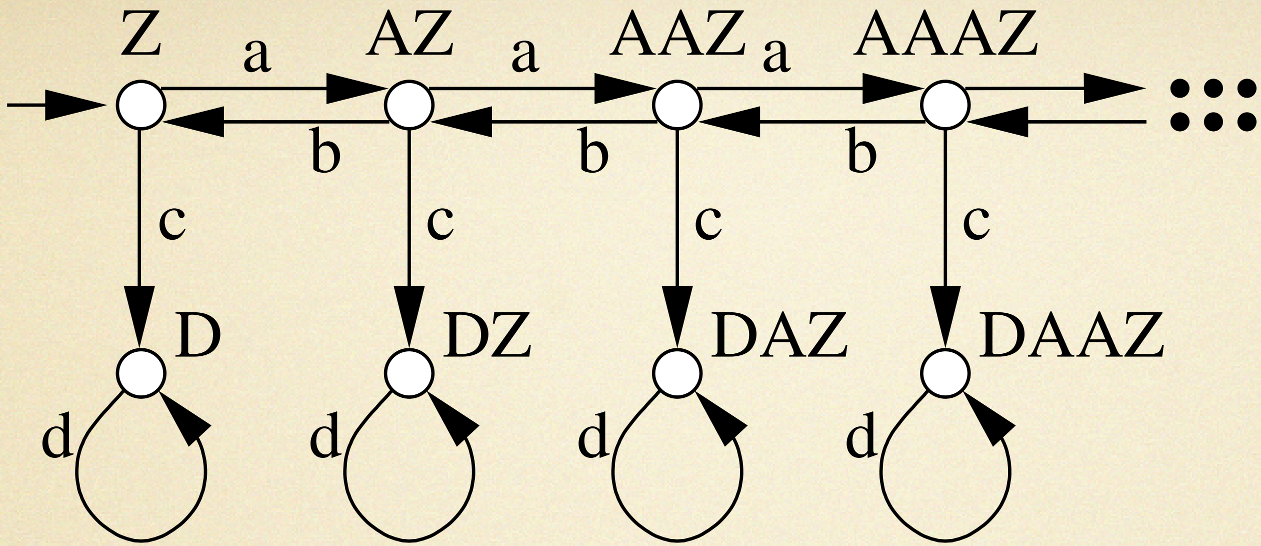
$D = d + uDD$






$$C = C \parallel C$$

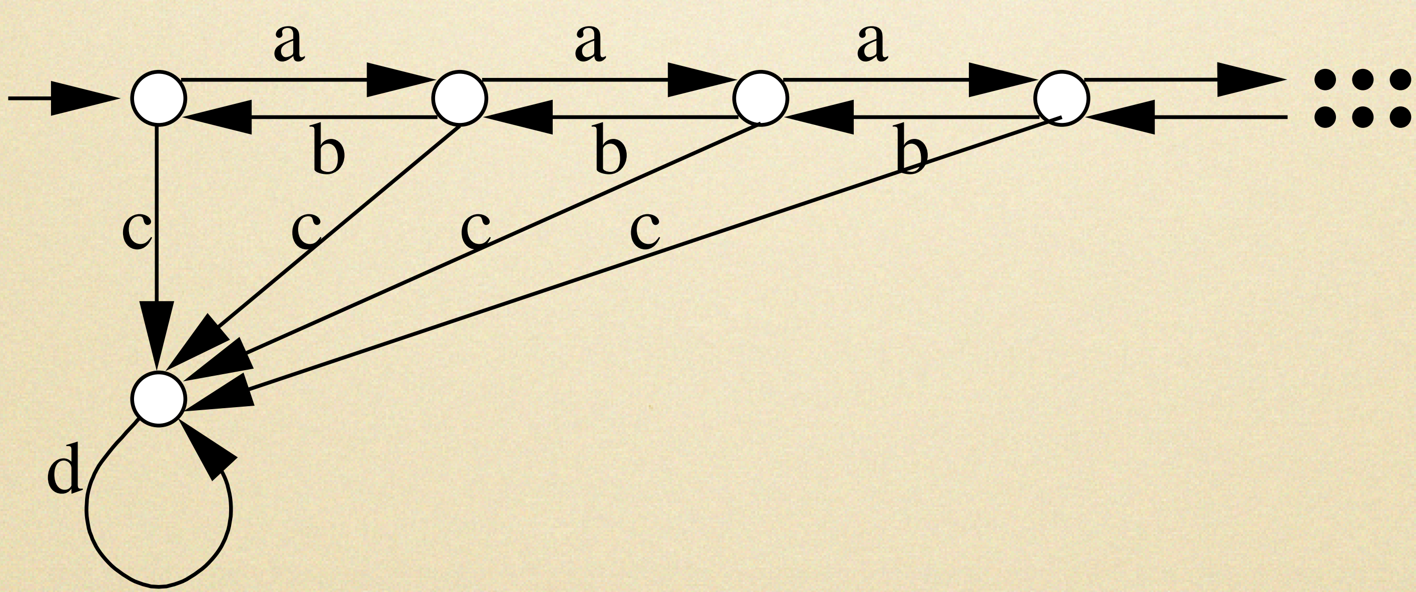



$Z = aAZ + cD$

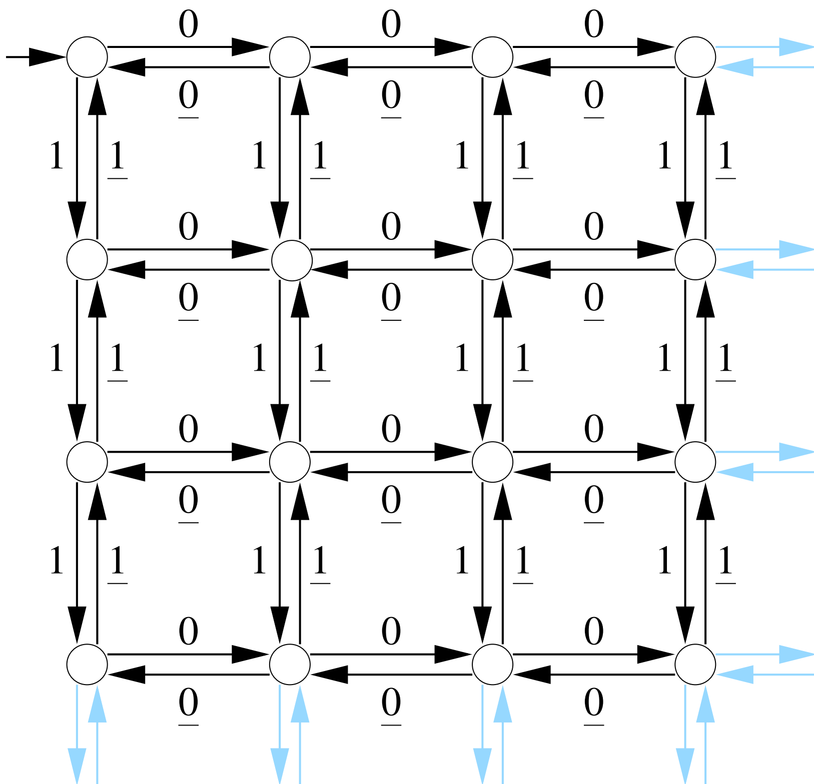
$A = aAA + cD + b$

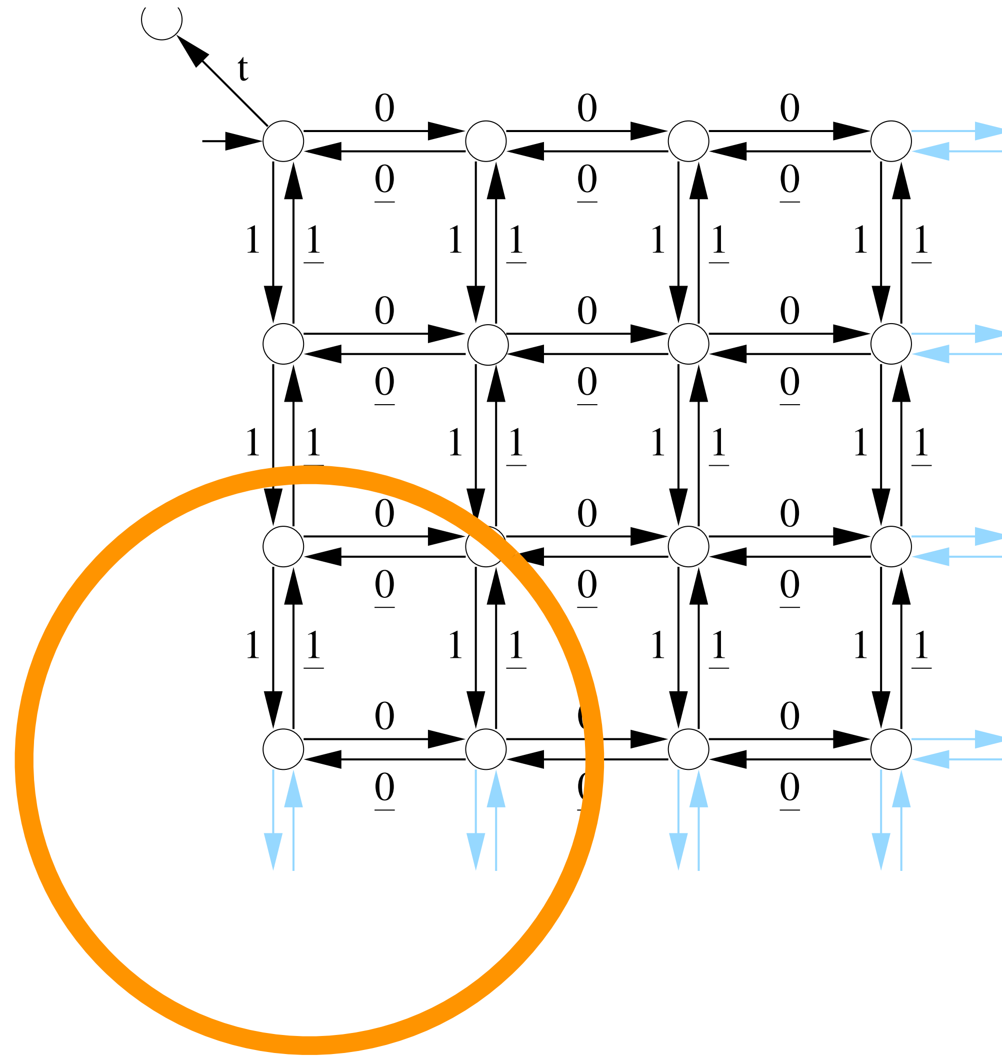
$D = dD$

not normed




**Theorem 1 (Caucal, 1990)** The class of normed BPA-graphs is closed under minimization.






*quadratic density*

*How to show for unnormed graphs that they are not BPA-definable?*

*Burkart, Caucal, Steffen: if g is a BPA graph, min(g) is a pattern graph*

*Caucal: pattern graphs of finite degree are context free graphs a la Muller and Schupp*

# Context-Free Graphs

**Definition.** A graph $\Gamma$ is context-free if $\Gamma$ is finitely generated

graph: (un)directed, labelled edges

finitely generated:
- connected, rooted
- uniformly bounded degree
- label alphabet is finite

such that

$$\{\Gamma(v) \mid v \text{ is vertex of } \Gamma\}$$

has only finitely many isomorphisms under end-isomorphisms.

---

$\Gamma$ a graph. For vertices $v$ of $\Gamma$:

$\Gamma(v)$ = connected component of $v$ in $\Gamma \setminus \Gamma^{(n)}$, where $n = |v|$;

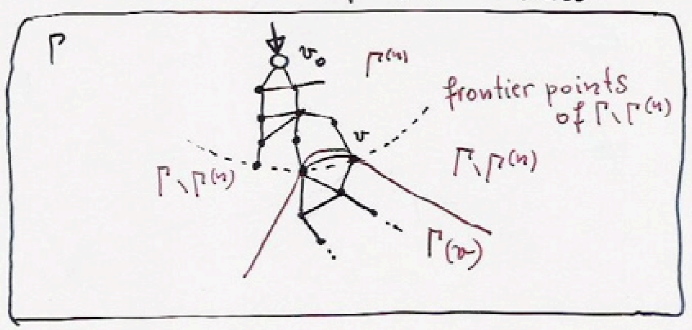
$|v|$ = distance of $v$ from the root of $\Gamma$;

$\Gamma^{(n)}$ = subgraph of $\Gamma$ consisting of all vertices and edges that are connected to $v_0$ by a path of length $< n$.

---

$\Delta(v)$ = set of frontier points in $\Gamma(v)$ (finite in fin. gen. graphs!)

a frontier-point of $\Gamma \setminus \Gamma^{(n)}$ is a vertex of $\Gamma \setminus \Gamma^{(n)}$ with $|u| = n$




---

$u, v$ vertices in $\Gamma$.

An end-isomorphism is a mapping $\varphi$ between $\Gamma(u)$ and $\Gamma(v)$ such that

(i) $\varphi$ is a label-preserving graph-isomorphism

(ii) $\varphi$ maps $\Delta(u)$ onto $\Delta(v)$

---

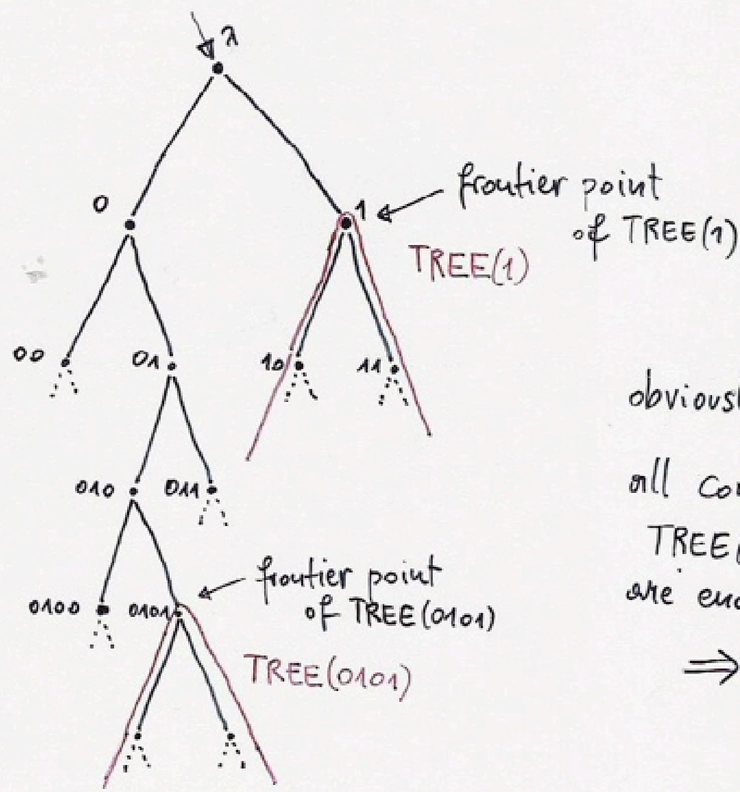

David E Muller, Paul E Schupp: "The theory of ends, pushdown automata and 2nd order logic, TCS 37 (1985) 51-75.

**Theorem.** A finitely generated graph $\Gamma$ is context-free if and only if $\Gamma$ is the (transition) graph of some pushdown automaton.

**Corollary.** If $\Gamma$ is context-free, then $\Gamma$ remains context-free with any vertex chosen as origin.

---

TREE

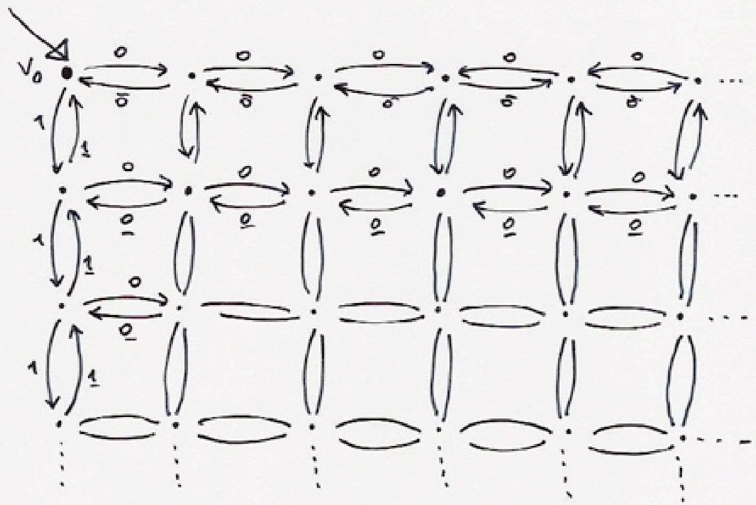



obviously:

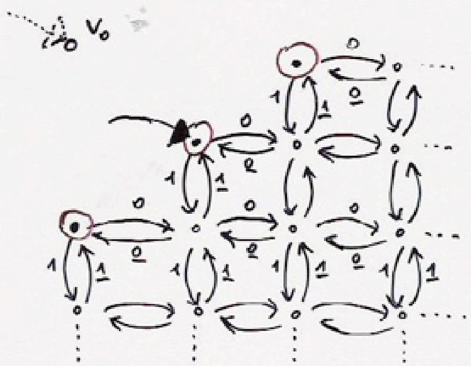
all components ("ends") TREE(u), TREE(v) are end-isomorphic

$\Rightarrow$ TREE is context-free.

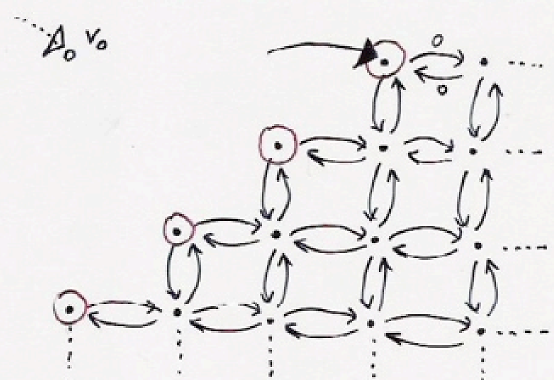
# BAG is not context-free




BAG




BAG((-1,-1))




BAG((0,3))

(o ... frontier point)

are not end-isomorphic segments ("ends")

$\Longrightarrow$ BAG is not context-free !

$$\mathrm{Q} = \mathrm{Q}_\lambda = \sum_{d \in D} \mathrm{r}_1(\mathrm{d}) \cdot \mathrm{Q}_{\mathrm{d}}$$
$$\mathrm{Q}_{\sigma \mathrm{d}} = \mathrm{s}_2(\mathrm{d}) \cdot \mathrm{Q}_\sigma + \sum_{e \in D} \mathrm{r}_1(\mathrm{e}) \cdot \mathrm{Q}_{\mathrm{e}\sigma\mathrm{d}}$$
(for $d \in D$, and $\sigma \in D^*$)

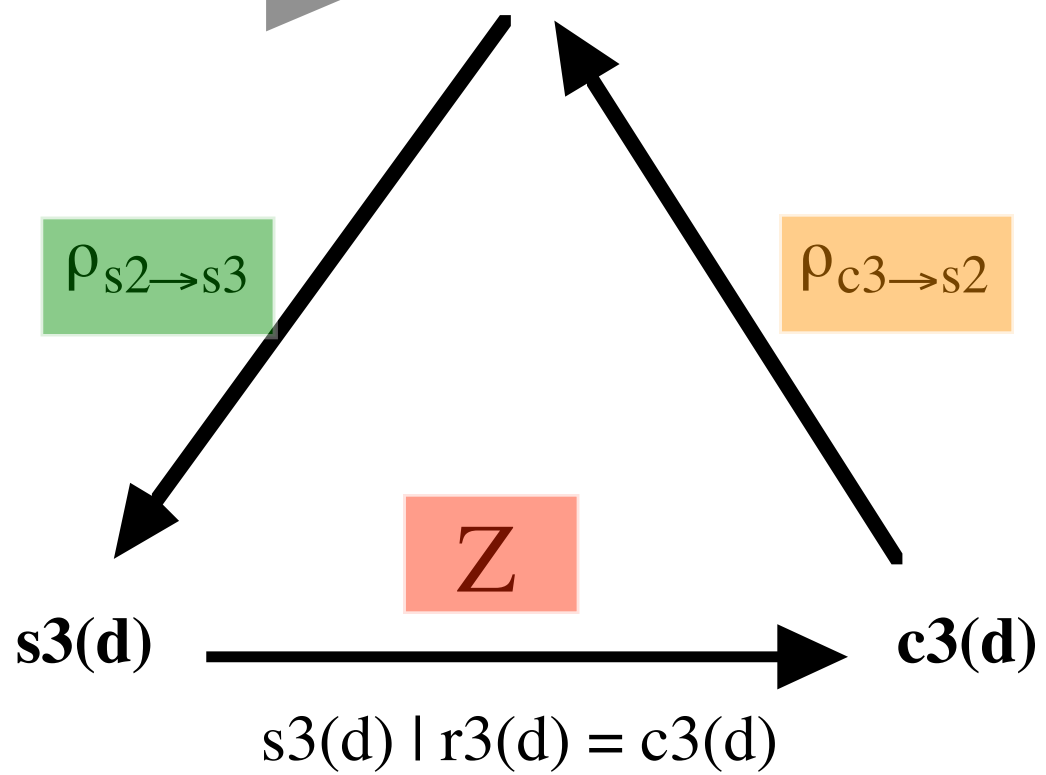
Table 2: Queue, infinite BPA-specification

$$\mathrm{Q} = \sum_{d \in D} \mathrm{r}_1(\mathrm{d})(\rho_{\mathrm{c}_3 \to \mathrm{s}_2} \circ \partial_H)(\rho_{\mathrm{s}_2 \to \mathrm{s}_3}(\mathrm{Q}) \parallel \mathrm{s}_2(\mathrm{d}) \cdot \mathrm{Z})$$
$$\mathrm{Z} = \sum_{d \in D} \mathrm{r}_3(\mathrm{d}) \cdot \mathrm{Z}$$

Table 3: Queue, finite ACP-specification with renaming

$$Q = \Sigma_{d\in D}\, \overset{\text{guard}}{r1(d)}(\rho_{c3\rightarrow s2} \circ \partial_H)(\rho_{s2\rightarrow s3}(Q) \| s2(d).Z)$$

$$Z = \Sigma_{d\in D}\, r3(d).Z$$


actions: r1(d), s2(d)

auxiliary actions: r3(d), s3(d), c3(d)

communication: r3(d) | s3(d) = c3(d)

$\rho_{s2\rightarrow s3}$ renaming: s2(d) → s3(d)

$\rho_{c3\rightarrow s2}$ renaming: c3(d) → s2(d)

encapsulation: $H = \{s3(d), r3(d) \mid d \in D\}$

Bergstra-Tiuryn:

Queue cannot be defined in ACP with handshaking communication

- but it can in ACP with renaming,

- or in ACP with ternary communication

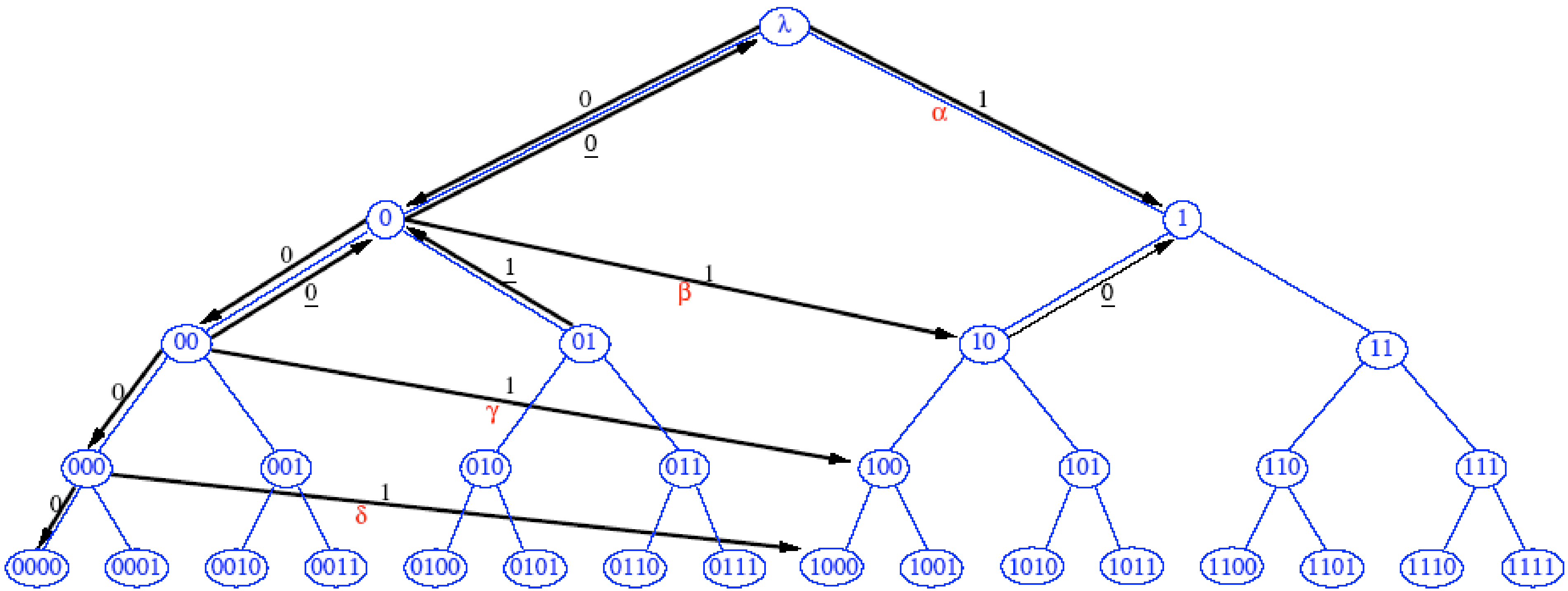
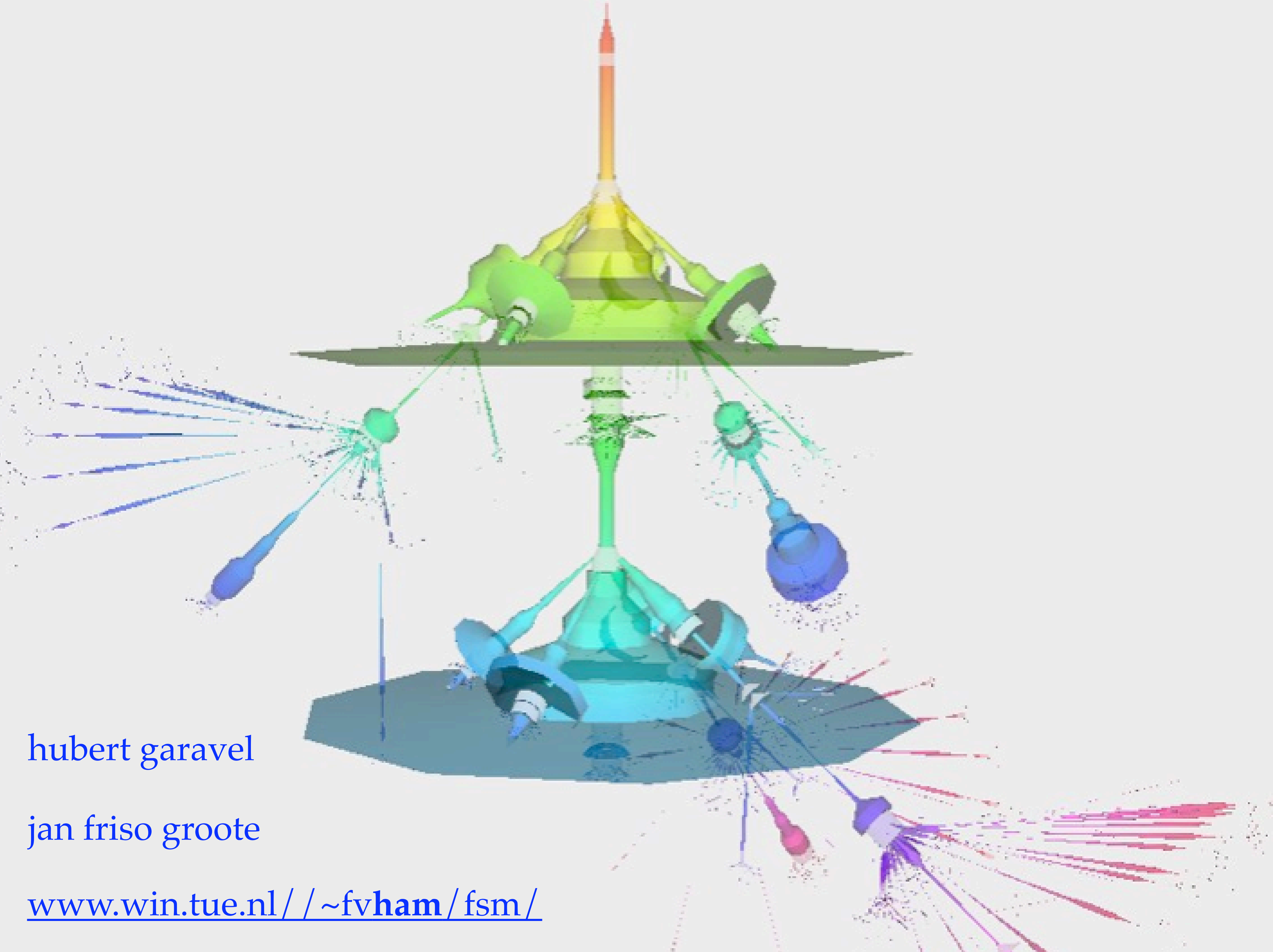



Figure 6: Attempt at drawing Queue in ‘tree space’.

*Science fiction*

*can we derive properties from the topology or geometry of process graphs of large state spaces?*



hubert garavel

jan friso groote

[www.win.tue.nl//~fv**ham**/fsm/](www.win.tue.nl//~fvham/fsm/)