Reflections on a Geometry of Processes

Clemens Grabmayer* Jan Willem Klop† Bas Luttik‡

November 30, 2005

Abstract

In this note we discuss some issues concerning a geometric approach to process algebra. We mainly raise questions and are not yet able to present significant answers.

1 Periodic Processes

Our point of departure is the axiom system BPA in Table 1 together with guarded recursion.

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

Table 1: BPA (Basic Process Algebra)

We are in particular interested in *non-linear* recursion, where products of recursion variables are allowed, in contrast with linear recursion exemplified by $\langle \mathrm{X} | \mathrm{X} = \mathrm{aY} + \mathrm{b}, \mathrm{Y} = \mathrm{cX} + \mathrm{dY} \rangle$ yielding only regular (finite-state) processes. Non-linear recursion also allows infinite-state processes, such as the counter $\langle \mathrm{C} | \mathrm{C} = \mathrm{uDC}, \mathrm{D} = \mathrm{uDD} + \mathrm{d} \rangle$ (with actions u, d for "up" and "down") or the process Stack that is definable by the infinite set of linear recursion equations over BPA (cf. the left-hand side of Table 2), and more remarkably, by the finite set of non-linear recursion equations (cf. the right-hand side of Table 2).

This simple framework is already rich in structure. In [1] this framework was linked with context-free grammars (CFG's), in particular with those in (restricted) Greibach normal form.

*Vrije Universiteit Amsterdam. Postal address: Department of Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands. E-mail: `clemens@cs.vu.nl`.

†Vrije Universiteit Amsterdam, Radboud Universiteit, and CWI. Postal address: Department of Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands. E-mail: `jwk@cs.vu.nl`.

‡Eindhoven Technical University and CWI. Postal address: P.O. Box 513, 5600 MB Eindhoven, The Netherlands. E-mail: `s.p.luttik@tue.nl`.

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

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

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

There the fact was established that while the language equality problem for CFG's is unsolvable, the process equality problem for CFG's is solvable. A priori this is not implausible, because a process has much more inner 'structure' than a language (the set of its finite terminating traces). The decidability was demonstrated by Baeten, Bergstra, and Klop in [1] as a corollary of a result concerning the periodical geometry or topology of the corresponding process graph. In Figure 1 the periodicities of two examples are exhibited: of Stack on the left-hand side, and of the process $\langle X | X = bY + dZ, Y = d + dX + bYY, Z = b + bX + dZZ \rangle$ on the right-hand side (this graph repeats three finite graph fragments α, β and γ as is also illustrated in Figure 2 below).

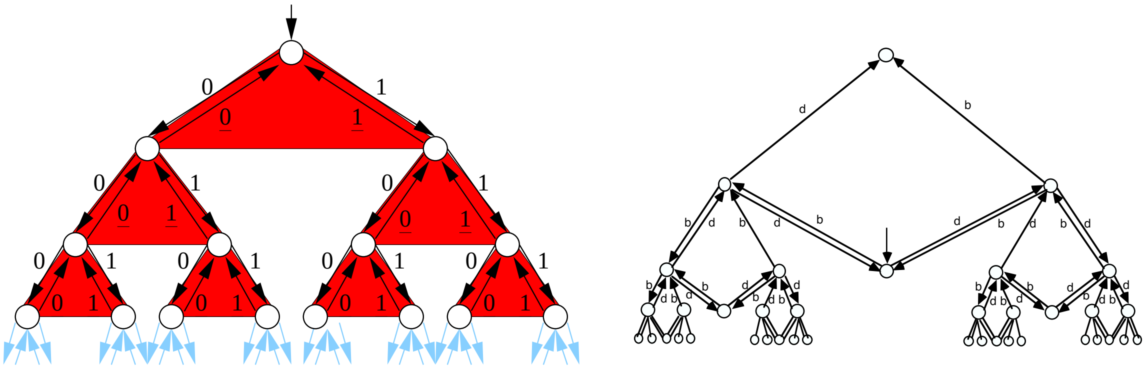

Figure 1: Tree-like periodic processes

The geometric proof in [1] is complicated. For the corollary of the decidability more stream-lined approaches have subsequently been found by using tableaux methods and other arguments (cf. Caucal in [7], Hüttel and Stirling in [11], and Groote in [10]). Also, the geometric aspects have been studied, for example by Caucal in [8] and by Burkart, Caucal, and Steffen in [5]. Actually, the related notion of *context-free graph* was introduced by Muller and Schupp [13] already in 1985.

We feel that there is still much to be explained about the geometric aspects of process graphs. We present a question concerning the fact that periodic graphs in BPA come in two kinds: 'linear' graphs as on the left-hand side, and 'branching' graphs as on the right-hand side in Figure 2.

Question 1 *Is it decidable whether a system E of equations (in Greibach normal form) yields a linear (type I) or a branching (type II) graph?*

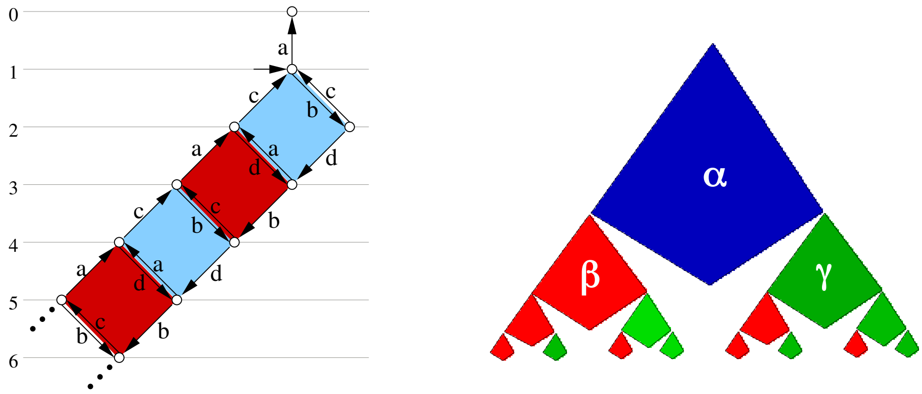

Figure 2: ‘Linear’ periodic graphs (type I, left), ‘branching’ periodic graphs (type II, right)

Another graph of type II is the ‘butterfly’ process graph in Figure 3 of the recursive BPA-specification $\langle X|X = a + bY + fXY,\ Y = cX + dZ,\ Z = gX + eXZ\rangle$. The relevance of the

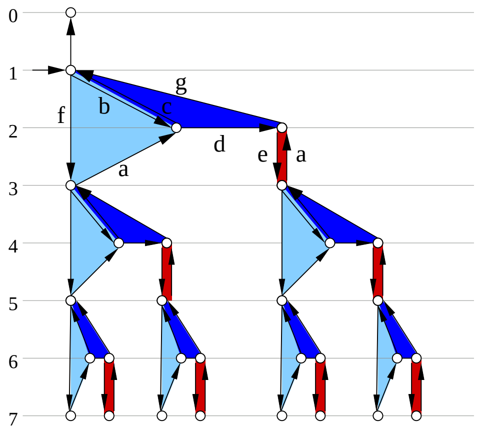

Figure 3: A ‘butterfly’ process graph.

distinction between type I and type II graphs is made clear below, in order to show that certain graphs are *not* of type I or type II.

In the study of BPA-definable graphs an important property is that of being “normed”. A graph is *normed* if from every node in it there is a path to a terminating node. (In term rewriting terminology this is called the weak normalization property WN.) The norm of a node is then the minimum number of steps to termination. Originally, the decidability of context-free processes (BPA-definable processes) was established in [1] only for the normed case. Subsequently this was generalized by Christensen, Hüttel, and Stirling in [9] to all BPA-definable processes.

Note that the norm of a node in a process graph is preserved under bisimulation: if norms are pictorially represented by drawing the process graph with horizontal ‘level’ lines, arranging points with the same norm on the same level (see the graph left in Figure 2 and the graph in Figure 3), then bisimulations relate only points on horizontal lines. Collapsing a normed graph to its canonical form is a compression in horizontal direction.

An important question is whether BPA-definable processes are closed under minimization (i.e. under compressing a graph such that it is minimal under bisimulation; the resulting graph is also called the "canonical" graph). The question whether such a statement does in fact hold was left open in [1]. Making a graph canonical can alter its geometry considerably. For instance, consider the counter C mentioned above. The process graph *g* of C is a linear sequence of nodes $\mathrm{C}, \mathrm{DC}, \mathrm{DDC}, \ldots$ connected by u-steps to the right and d-steps to the left. The merge $\mathrm{C} \parallel \mathrm{C}$ in the process algebra PA has a grid-like graph similar to that of the process Bag on the left side in Figure 6 below. But if we collapse this graph *g* for $\mathrm{C} \parallel \mathrm{C}$ to its canonical form by identifying the bisimilar nodes on diagonal lines, we obtain again the graph *g* for C. So a grid may collapse to a linear graph.

Normedness plays a part when graphs are compressed to their canonical form. In [5] Burkart, Caucal, and Steffen give the following example of a BPA-graph that after compression to canonical form no longer is a BPA-graph: For the process with recursive definition $\langle \mathrm{Z} | \mathrm{Z} = \mathrm{aAZ} + \mathrm{cD}, \mathrm{A} = \mathrm{aAA} + \mathrm{cD} + \mathrm{b}, \mathrm{D} = \mathrm{dD} \rangle$ in BPA, the graph on the left in Figure 4 is its associated BPA-process graph, while the graph on the right is the respective minimization,

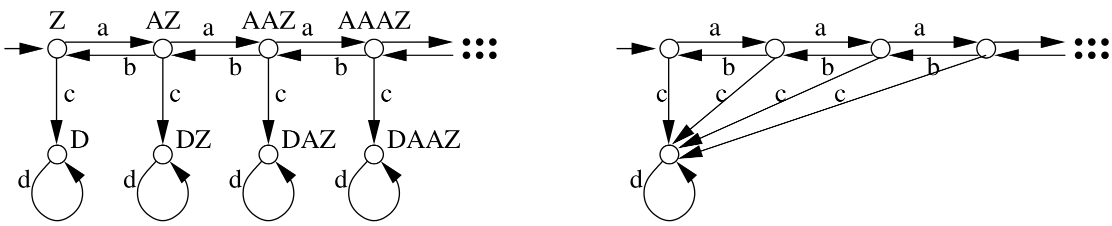

Figure 4: Counterexample against the preservation of BPA-graphs under minimization.

which does not have the periodical structure of a BPA-graph. Note that neither of these graphs is normed.

Question 2 *How can those BPA-graphs be characterized whose canonical graphs are again* BPA-*graphs?*

We note that Question 2 has already received quite some attention in Caucal's work. Contrasting with the counterexample for the unnormed case given above, in [7] he has shown the following theorem.

Theorem 3 (Caucal, 1990) The class of normed BPA-graphs is closed under minimization.

The (obvious) link between CFG's and BPA-definable processes was first mentioned in [1]. An example is the graph on the right in Figure 1 and in Figure 2 above: it determines as context-free language (CFL) the language of words having equal numbers of letter b and d. An intriguing question is the following.

Question 4 *How does the classical pumping lemma for CFL's relate to the periodicity present in* BPA-*definable processes?*

Another interesting observation, due to H.P. Barendregt, is the following. It is well-known that the language $L = \{a^n b^n c^n | n \geq 0\}$ is not a CFL. This language can be obtained as the set of finite traces of the triangular, infinite, minimal graph in Figure 5. Intuitively it is obvious that this graph is not tree-like periodic. This leads to the next question.

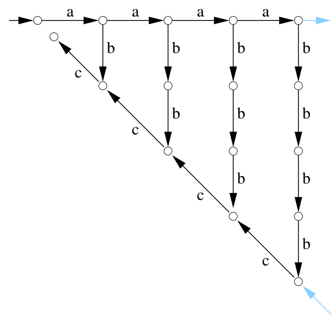

Figure 5: The language L.

Question 5 *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?*

2 Non-definability of Bag in BPA

The expressiveness of the operations defined by the axioms of BPA is limited; basically only sequential processes can be defined. The axiom system PA is an extension of BPA with axioms for the merge $\|$ (interleaving) and the auxiliary operator $\lfloor\!\lfloor$ (left merge). In PA we

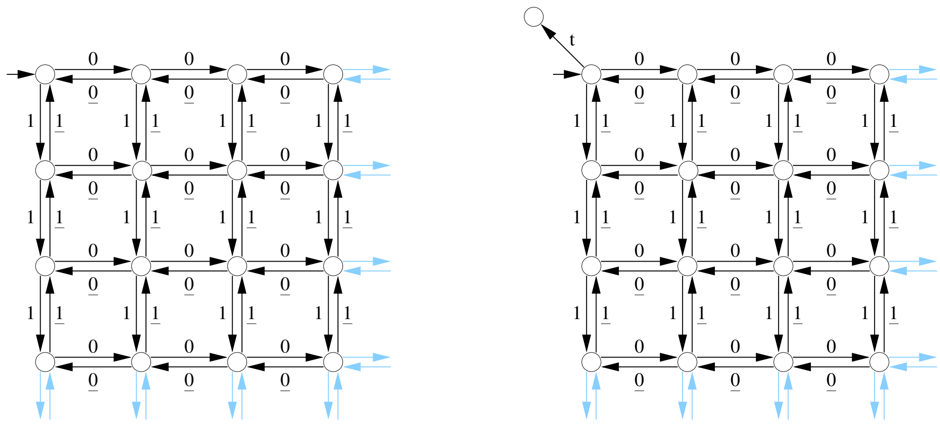

Figure 6: The minimal process graphs of the process Bag (on the left-hand side), and of a terminating variant $\mathrm{Bag_t}$ of Bag (on the right-hand side).

have a succinct recursive definition for the process Bag (over data $\{0,1\}$) as follows:

$$\mathrm{B} = 0(\underline{0} \,\|\, \mathrm{B}) + 1(\underline{1} \,\|\, \mathrm{B}).$$

It has been proved by Bergstra and Klop in [3] that the process Bag cannot be defined by means of a finite recursive specification over BPA. Considering the minimal process graph for it in Figure 6, this does not come as a surprise: it is not tree-like, but "grid-like". Below we give an alternative proof of this fact.

Theorem 6 (Bergstra, Klop, 1984) Bag is not BPA-definable.

Proof (Sketch). Suppose that the process Bag is BPA-definable. Then there exists a recursive specification E in BPA such that Bag is bisimilar with a tree-like periodic graph $g(E)$ as defined by Baeten, Bergstra, and Klop in [1]. Then $g(E)$ is a "BPA-graph" according to the terminology used in [5].[1]

In [5] Burkart, Caucal, and Steffen have shown that, for *every* BPA-graph G, the canonical graph of G is a "pattern graph", which means that it can be generated from a finite (hyper)graph by a reduction sequence of length ω according to a deterministic (hypergraph) grammar.[2] Since Bag is itself a canonical graph and since therefore Bag is the canonical graph of the BPA-graph $g(E)$, it follows that Bag is a pattern graph.

A theorem due to Caucal in [8] states that all (rooted) pattern graphs of finite degree are "context-free" according to the definition of Muller and Schupp in [13].[3] It follows that Bag is context-free. However, it is not difficult to verify that Bag is actually *not* a context-free graph according to the definition in [13].

In this way we have arrived at a contradiction with our assumption that Bag is definable in BPA. □

By using Caucal's theorem, Theorem 3, it is also possible to establish quickly the non-definability in BPA of many normed graphs. For example, for the terminating version Bag_t of Bag (where Bag_t is normed) with the process graph on the right in Figure 6, it can be reasoned as follows. This graph is canonical, so if it were BPA-definable, then it would be a graph of type I or type II. However, for a type I graph it holds that the number of nodes in a sphere $B(s,\rho)$, where s is the center and ρ is the radius, depends linearly on ρ; for a type II graph this dependance is of exponential form. But for the graph under consideration the number of nodes in a ball $B(s,\rho)$ only depends quadratically on ρ. Hence this graph is not BPA-definable.

Where do we need the preservation of BPA-definability under minimization? The process graph of Bag_t is clearly not one obtainable by a BPA-definition, as it is not of type I or type II. But equality of processes is considered here modulo bisimulation—so it is not inconceivable that there is a BPA-definition E of Bag_t such that $g(E)$ *after compression* to canonical form $\mathrm{can}(g(E))$ were just the process graph $\mathrm{graph}(\mathrm{Bag}_\mathrm{t})$ for Bag_t on the right in Figure 6. So $\mathrm{can}(g(E)) = \mathrm{graph}(\mathrm{Bag}_\mathrm{t})$ holds. But with the preservation property, Theorem 3, we have $\mathrm{can}(g(E)) = g(E')$ for some BPA-specification E', hence $g(E')$, and therefore $\mathrm{graph}(\mathrm{Bag}_\mathrm{t})$, are of type I or type II, quod non.

[1] In earlier papers of Caucal (e.g. in [6] and [8]) BPA-graphs were known under the name "alphabetic graphs".

[2] "Pattern graphs" according to this definition used by Caucal and Montfort in [6] are called "regular graphs" in the later paper [5] by Burkart, Caucal, and Steffen. Because the use of the attribute "regular" for process graphs could lead to wrong associations, we avoid this terminology from (hyper)graph rewriting here.

[3] Note that the class of "context-free" graphs in Muller and Schupp's definition does not coincide with the graphs associated with "context-free" processes (the class of BPA-graphs), but that it forms a strictly richer class of graphs corresponding to the class of transition graphs of push-down automata.

3 The strange geometry of Queue

After the paradigmatic processes Stack and Bag, we now turn to the third paradigmatic process Queue (the first-in-first-out version with unbounded capacity). Table 3 gives the infinite BPA-specification.

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

Table 3: Queue, infinite BPA-specification

As before, the endeavour is to specify Queue in a finite way. It was proved by Bergstra and Tiuryn [4] that the system BPA is not sufficient for that; in fact, they showed that Queue cannot even be defined in ACP *with handshaking communication* (see [2] for a complete treatment of the axiom system ACP). But Queue has a finite recursive specification in ACP with *renaming* operators (see Table 4, the specification is originally due to Hoare using the 'chaining'-operation).

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

Table 4: Queue, finite ACP-specification with renaming

An ambitious question is the following.

Question 7 *Is there a geometric (topological) property of processes definable by handshaking communication?*

Finally, we turn to geometric properties of the process Queue. Surprisingly, it is unexpectedly problematic to draw the process graph of Queue in a 'neat' way (cf. also Figure 7), similar to Stack and Bag. We would like to uncover the 'deep' reason for this difficulty.

Question 8 *Is it possible to fit* $g(\mathrm{Queue})$ *in the binary tree space?*

References

[1] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Decidability of bisimulation equivalence for process generating context-free languages. *Journal of the ACM*, 40(3):653–682, 1993.

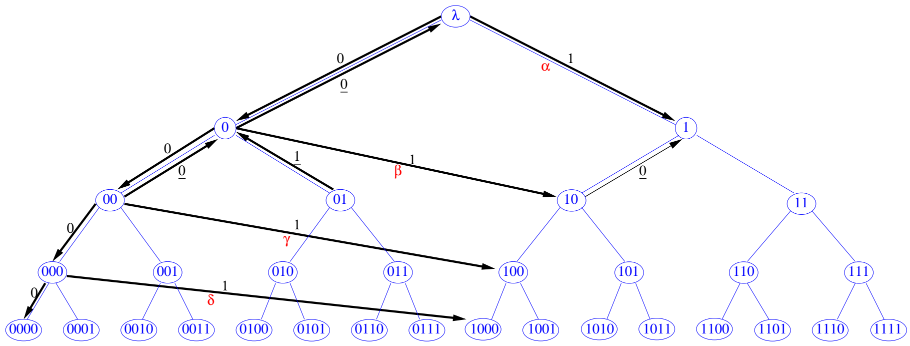

Figure 7: Attempt at drawing Queue in ‘tree space’.

[2] J. C. M. Baeten and W. Peter Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[3] J. A. Bergstra and J. W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In J. Paredaens, editor, *Proceedings of ICALP’84*, volume 172 of *LNCS*, pages 82–95. Springer, 1984.

[4] J. A. Bergstra and J. Tiuryn. Process algebra semantics for queues. *Fundamenta Informaticae*, X:213–224, 1987.

[5] O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. In *Proceedings of CONCUR’96*, 1996.

[6] D. Caucal and R. Montfort. On the transition graphs of automata and grammars. In *Proceedings of WG 90*, volume 484 of *LNCS*, pages 61–86. Springer, 1990.

[7] D. Caucal. Graphes canoniques de graphes algébriques. *Theoret. Inform. and Appl.*, 24(4):339–352, 1990.

[8] D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 1992.

[9] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.

[10] J. F. Groote. A short proof of the decidability of bisimulation for normed bpa-processes. *Information Processing Letters*, 42:167–171, 1992.

[11] H. Hüttel and C. Stirling. Actions speak louder than words: Proving bisimilarity for context-free processes. In *Proceedings of LICS’91*, pages 376–386, 1991.

[12] P. Linz. *An introduction to formal languages and automata*. Jones and Bartlett Publishers, third edition, 2001.

[13] D.E. Muller and P.E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 1985.