Closing the Image of the Process Interpretation of 1-Free Regular Expressions Under Bisimulation Collapse

Clemens Grabmayer
Gran Sasso Science Institute, L'Aquila, Italy
clemens.grabmayer@gssi.it

This extended abstract reports on a refinement of results on the structure of Milner's process interpretation of regular expressions, when restricted to 1-free expressions with binary star, that Fokkink and myself used to solve Milner's axiomatization question for 1-free regular expressions. Here we transfer the property of preservation–under–bisimulation–collapse from a structural condition of expressible process graphs (the layered loop existence and elimination property LLEE) to the image of the process interpretation of regular expressions with unary star, albeit to a compact variant of the process interpretation and restricted to 'under-star-1-free' expressions. This technical result highlights a noteworthy feature of the process semantics in the context of that, as we showed earlier, the image of the (unrestricted) compact process interpretation is not closed under bisimulation collapse.

In Section 1 we explain the process interpretation of regular expressions, and summarize the results for interpretations of '1-free' expressions with binary star obtained in [11]. Then in Section 2 we define the compact process interpretation by using a shortcoming of the process interpretation as motivation. Finally in Section 3 we explain the steps by which our main result is obtained via refined extraction of regular expressions from process graphs with the layered loop existence and elimination property LLEE.

1 Introduction (the process interpretation of 1-free regular expressions)

In [13], Milner introduced a process interpretation P for regular expressions with constant 0, letters over a given set A, and as regular operators the binary operators $+$ and $\cdot$, and the unary operator star $(\cdot)^*$. Informally, the process interpretation $P(e)$ of a regular expression e is defined by the following inductive clauses: 0 is interpreted as a deadlocking process without any observable behavior, letters from the set A stand for atomic actions that lead to successful termination; the binary operators $+$ and $\cdot$ are interpreted as the operations of choice and concatenation of two processes, respectively, and the unary star operator $(\cdot)^*$ is interpreted as the operation of unbounded iteration of a process, but with the option to terminate successfully before each iteration. We also add the constant 1 whose behavior is stipulated to be that of 0^*, the immediately terminating process known as ε in process theory. Formally, Milner used this intuition to define $P(e)$ by induction on the structure of regular expressions e as labeled transition graphs with an immediate-termination predicate $\Downarrow$. Based on the so-defined process interpretation P, Milner then defined the process semantics $[\![e]\!]_P$ of a regular expression e as the behavior of $P(e)$, and that is as the bisimilarity $\leftrightarrows$ equivalence class $[P(e)]_{\leftrightarrows}$ of the labeled transition graph $P(e)$.

The process interpretation $P(e)$ of a regular expression e refines the language semantics $[\![e]\!]_L$ of e (defined first by Copi, Elgot, Wright [3]) in the sense that traces to termination of the process $P(e)$ correspond to words of the language $[\![e]\!]_L$. However, different from the language semantics $[\![\cdot]\!]_L$, in which every regular language L (accepted by a finite-state automaton) is obtained as the language semantics $L = [\![e]\!]_L$ of some regular expression e, the process semantics $[\![\cdot]\!]_P$ is incomplete: not every finite process graph (corresponding to a non-deterministic finite-state automaton) is *($[\![\cdot]\!]_P$-)expressible* and that is, bisimilar

Submitted to:
TERMGRAPH 2024

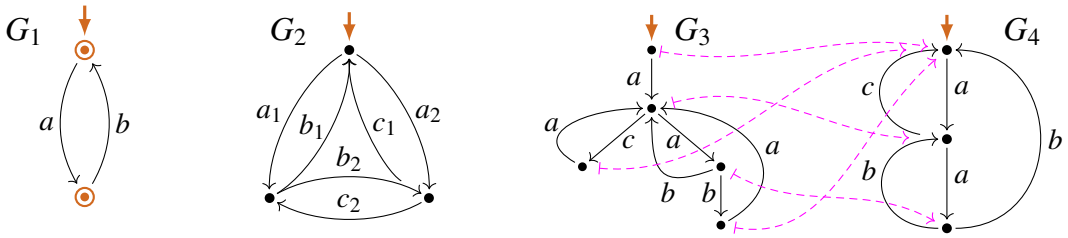

Figure 1: The process graphs G_1 and G_2 are not $[\![\cdot]\!]_P$-expressible. G_3 is P- and $P_{\circledast}$-expressible as $G_3 = P((a\cdot(c\cdot a + a\cdot(b + b\cdot(a+a)))^*)\cdot 0) = P_{\circledast}(a\cdot(\ldots)^{\circledast}0)$. So G_3 is $[\![\cdot]\!]_P$-expressible, which then also holds for its bisimulation collapse G_4, the image of G_3 via the indicated functional bisimulation. (Start vertices are indicated by a brown arrow ➝, vertices with immediate termination by ◉ with a brown ring.)

to the process interpretation of some regular expression. For example, the collapsed graphs G_1 and G_2 in Fig. 1, are not $[\![\cdot]\!]_P$-expressible as shown by Bosscher [2] and Milner [13]. In contrast, G_3 in Fig. 1 is P- and hence $[\![\cdot]\!]_P$-expressible graph, and thus its collapse G_4 there is $[\![\cdot]\!]_P$-expressible collapse G_4 in Fig. 1 (we will recognize G_4 as P-expressible later in Fig. 3). The failure of completeness led Milner to pose the question of how the structure of $[\![\cdot]\!]_P$-expressible finite process graphs can be characterized. Although the underlying *expressibility problem* is (difficult but) decidable [1], an answer to Milner's characterization question is complicated by the fact that bisimilarity can significantly alter the structure of process graphs. It is easier to investigate the structure of *P-expressible* graphs that are isomorphic to ones in the image of the process interpretation P, and within those to focus on bisimulation collapsed graphs.

Yet even the structure of P-expressible graphs is difficult to capture, because also the image of P is 'incomplete': it is not closed under bisimilarity, and not even under bisimulation collapse (see [8, 10]). This notwithstanding, P can express directly, via isomorphism, many process graphs that exhibit compact forms of sharing. Indeed, Fokkink and I have identified a structural property of process graphs [11], *the (layered) loop existence and elimination property (L)LEE*,[1] that can act as a useful proxy for P-expressibility. We established a close link between LLEE and process interpretations of '1-free' regular expressions with binary star $\circledast$ (see $\mathbf{(I)}^{(+)}_{P_{\circledast}}$, $\mathbf{(E)}^{(+)}_{P_{\circledast}}$ below), and showed that LLEE is preserved under bisimulation collapse (see $\mathbf{(C)}_{\mathrm{st}}$ below). For this purpose, we used a formulation $P_{\circledast}$ of the process interpretation for regular expressions with binary star $\circledast$ (as originally used by Kleene [12] for 'regular events'). Informally the process interpretation $P_{\circledast}(e_1{}^{\circledast}e_2)$ of a binary-iteration expression $e_1{}^{\circledast}e_2$ denotes the process that starts with unbounded iteration of the process denoted by e_1, and, after termination, continues with the process denoted by e_2. The formal definition of $P_{\circledast}$ in [11] interprets regular expressions e with binary star as finite *sink-termination* process graphs $P_{\circledast}(e)$, which contain at most a single vertex with immediate termination that, furthermore, cannot be the start vertex. For 1-free regular expressions e with binary star and their process interpretation $P_{\circledast}(e)$, we showed the following statements in [11]:

$\mathbf{(I)}^{(+)}_{P_{\circledast}}$: Process ***i***nterpretations $P_{\circledast}(e)$ of 1-free regular expressions e with unary binary star $\circledast$ are finite sink-termination process graphs that satisfy LLEE.

$\mathbf{(E)}^{(+)}_{P_{\circledast}}$: From every finite sink-termination process graph G with LLEE a 1-free regular expression e with binary star $\circledast$ can be ***e***xtracted whose process interpretation $P_{\circledast}(e)$ is bisimilar to G.

$\mathbf{(C)}_{\mathrm{st}}$: The class of finite sink-termination process graphs with LLEE is closed under bisimul. ***c***ollapse.

$\mathbf{(Exp)}^{(+)}_{P_{\circledast}}$: A finite sink-termination process graph is ***exp***ressible by a 1-free regular expression with binary star $\circledast$ if and only if its bisimulation collapse satisfies LLEE.

[1] Layered LEE (LLEE) is formally stronger than, but equivalent to, LEE. Yet LLEE is more expedient for proofs than LEE.

The formulation of these results for regular expressions with binary star was motivated by two (related) reasons: (i) for regular expressions with binary star $\circledast$ the condition to be '1-free' has a clear-cut meaning because the empty process ε is not definable if the constant 1 (which we use for ε) is absent (unlike definability of 1 by 0^* with unary star), and (ii) interpreting only 1-free regular expressions with binary star entails a significant simplification of the topological structure of the obtained process graphs, namely that LLEE holds, see $\mathbf{(I)}^{(+)}_{P_\circledast}$, which is not the case in general (as witnessed by easy counterexamples in [7]).

2 Compact process interpretation (motivation and definition)

The statements $\mathbf{(I)}^{(+)}_{P_\circledast}$ and $\mathbf{(E)}^{(+)}_{P_\circledast}$ above from [11] have established a close link (but not a bijective correspondence) between the image of the process interpretation P, when restricted to 1-free regular expressions with binary star, and the property LLEE of process graphs. Now since LLEE is closed under bisimulation collapse due to $\mathbf{(C)}_{\mathrm{st}}$, the question arises of whether the image of P is also closed under bisimulation collapse at least for the restriction to 1-free expressions. While we find here that the answer is negative for P, we also obtain a positive answer for a slight variation $P^\bullet$ of P that increases sharing and produces more compact process graphs. But we know from [8, 10] that this result cannot be extended to the full image of $P^\bullet$. In this section we motivate and define the compact process interpretation $P^\bullet$.

In order to relate our results directly to formulations of the process interpretation with unary star (like Milner's), we also define '1-free' regular expressions with unary star, as interpretations of regular expressions with binary star: every binary iteration subexpression $e_1{}^\circledast e_2$ is replaced by a subexpression $e_1^* \cdot e_2$. In doing so we keep in mind that the meaning of '1-free' regular expressions originates from expressions with binary star, and accept that expressions are constrained more than just by excluding 1 from regular expressions with unary star. We also define 'under-star-1-free' and normed regular expressions.

Definition 2.1 (regular expressions)**.** For every set A of actions, the set $RExp(A)$ of *regular expressions over* A, and its subsets $RExp^{(+)}(A)$, $RExp^{(+\backslash *)}(A)$, and *nd-*$RExp(A)$ of regular expressions over A that are 1-*free*, *under-star*-1-*free*, and *normed*, respectively, are defined by the following grammars:

$$\begin{array}{rlll} e,e_1,e_2 ::= & 0 \mid 1 \mid a \mid e_1+e_2 \mid e_1\cdot e_2 \mid e^* & (\text{where } a\in A) & RExp(A)\\ f,f_1,f_2 ::= & 0 \mid a \mid f_1+f_2 \mid f_1\cdot f_2 \mid (f_1)^*\cdot f_2 & (\text{where } a\in A) & RExp^{(+)}(A)\\ uf,uf_1,uf_2 ::= & 0 \mid 1 \mid a \mid uf_1+uf_2 \mid uf_1\cdot uf_2 \mid f^* & (\text{where } a\in A) & RExp^{(+\backslash *)}(A)\\ n,n_1,n_2 ::= & 1 \mid a \mid n+e \mid e+n \mid n_1\cdot n_2 \mid e^* & (\text{where } a\in A) & \textit{nd-}RExp(A) \end{array}$$

The following definition uses a transition system specification (TSS) for stipulating the process interpretation for regular expressions, a natural variant of Milner's definition that yields bisimilar graphs.

Definition 2.2 (process interpretation P)**.** For every regular expression $g \in RExp(A)$, *the process interpretation* $P(g)$ *of* g is defined as the (finite start-vertex connected) process graph $\langle V(g), A, g, \to, \Downarrow\rangle$ with vertices $V(g) \subseteq RExp(A)$, start vertex g, transition relation $\to \subseteq V(g)\times A\times V(g)$ and immediate termination relation $\Downarrow \subseteq V(g)$ that are defined via derivations in the transition system specification (TSS):

$$\frac{}{1\Downarrow} \qquad \frac{e_i\Downarrow}{(e_1+e_2)\Downarrow}\ (i\in\{1,2\}) \qquad \frac{e_1\Downarrow \qquad e_2\Downarrow}{(e_1\cdot e_2)\Downarrow} \qquad \frac{}{(e^*)\Downarrow}$$

$$\frac{}{a \xrightarrow{a} 1} \qquad \frac{e_i \xrightarrow{a} ed}{e_1+e_2 \xrightarrow{a} ed}\ (i\in\{1,2\}) \qquad \frac{e_1 \xrightarrow{a} ed}{e_1\cdot e_2 \xrightarrow{a} ed\cdot e_2} \qquad \frac{e_1\Downarrow \qquad e_2 \xrightarrow{a} ed}{e_1\cdot e_2 \xrightarrow{a} ed} \qquad \frac{e \xrightarrow{a} ed}{e^* \xrightarrow{a} ed\cdot e^*}$$

The interpretation and extraction statements $\mathbf{(I)}^{(+)}_{P_\circledast}$ and $\mathbf{(E)}^{(+)}_{P_\circledast}$ for regular expressions with binary star can be transferred to 1-free, and to under-star-1-free regular expressions with unary star. (For an explanation of the loop existence and elimination property LLEE we refer to [11, 4] and to the appendix.)

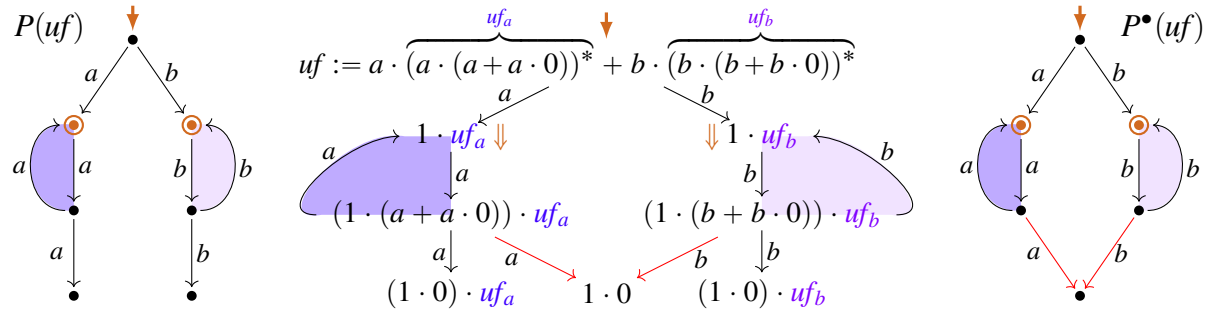

Figure 2: Example that highlights the reason why the process interpretation P cannot express all bisimulation collapses of interpretations of under-star-1-free regular expressions, while its compact process interpretation $P^\bullet$ can do so: For P, transitions that depart from different strongly connected components that are not nested (here colored indigo and darkcyan) carry trailing subexpressions of the different iterations that describe (also) the scc's (here uf_a and uf_b) although these iterations denote unreachable parts in the not normed target expressions at the bottom left and right. This does not happen for $P^\bullet$, because in it trailing unreachable parts are dropped from expressions that are not normed.

(I)$_P^{(\pm\backslash *)}$: Process ***i***nterpretations $P(uf)$ of under-star-1-free regular expressions $uf \in RExp^{(\pm\backslash *)}(A)$ are finite process graphs with actions in A that satisfy LLEE.

(E)$_P^{(\pm\backslash *)}$: From every finite process graph G with actions in A and with LLEE an under-star-1-free regular expression $uf \in RExp^{(\pm\backslash *)}(A)$ can be ***e***xtracted whose process interpretation $P_{\circledast}(uf)$ is bisimilar to G.

A shortcoming of the process interpretation P as defined above is that it prevents sharing in situations when an unnormed part like $g \cdot 0$ occurs jointly in separate and not nested iterations f_1^* and f_2^* of a regular expression e. Then due to the rules for products and iterations in the TSS in Def. 2.2, the parts of $P(e)$ that correspond to the occurrences of $g \cdot 0$ within f_1^* and f_2^* will carry along the iterations f_1^* and f_2^*, respectively, as unreachable parts, thereby preventing sharing of parts with the same behavior. A concrete example is illustrated in Fig. 2. This phenomenon can be remedied by refining two responsible rules for generating transitions in the TSS in Def. 2.2 in order to prevent propagating unreachable subexpressions.

Definition 2.3 (compact process interpretation $P^\bullet$)**.** For every regular expression $g \in RExp(A)$, *the compact process interpretation* $P^\bullet(g)$ *of* g is defined as the (finite, start-vertex connected) process graph $\langle V(g), A, g, \rightarrow, \Downarrow \rangle$ with vertices $V(g) \subseteq RExp(A)$, start vertex g, transition relation $\rightarrow \subseteq V(g) \times A \times V(g)$ and immediate termination relation $\Downarrow \subseteq V(g)$ that are defined via derivations in the TSS below that arises from the TSS in Def. 2.2 by replacing each of the third and the fifth rule for generating transitions by two versions, whose applicability depends on normedness of the underlying subexpression:

$$\frac{e_1 \xrightarrow{a} ed}{e_1 \cdot e_2 \xrightarrow{a} ed \cdot e_2}\ \text{(if } ed \text{ is normed)} \qquad \frac{e_1 \xrightarrow{a} ed}{e_1 \cdot e_2 \xrightarrow{a} ed}\ \text{(if } ed \text{ is not normed)}$$

$$\frac{e \xrightarrow{a} ed}{e^* \xrightarrow{a} ed \cdot e^*}\ \text{(if } ed \text{ is normed)} \qquad \frac{e \xrightarrow{a} ed}{e^* \xrightarrow{a} ed}\ \text{(if } ed \text{ is not normed)}$$

It is not difficult to show that $P^\bullet$ only increases sharing: by this we mean that $P(e) \rightrightarrows P^\bullet(e)$ holds for all regular expressions e, that is, that there is always a functional bisimulation from $P(e)$ to $P^\bullet(e)$. As a consequence, every $[\![\cdot]\!]_{P^\bullet}$-expressible process graph is also $[\![\cdot]\!]_P$-expressible, and vice versa.

We note that for the under-star-1-free regular expression uf in Fig. 2, the compact process interpretation $P^\bullet(uf)$ of uf is indeed a bisimulation collapse, different from its process interpretation $P(uf)$.

The interpretation and extraction statements **(I)**$_P^{(\pm\backslash *)}$ and **(E)**$_P^{(\pm\backslash *)}$ also hold for $P^\bullet$, and under-star-1-free regular expressions. We only formulate **(I)**$_{P^\bullet}^{(\pm\backslash *)}$ here, since we refine extraction in the next section.

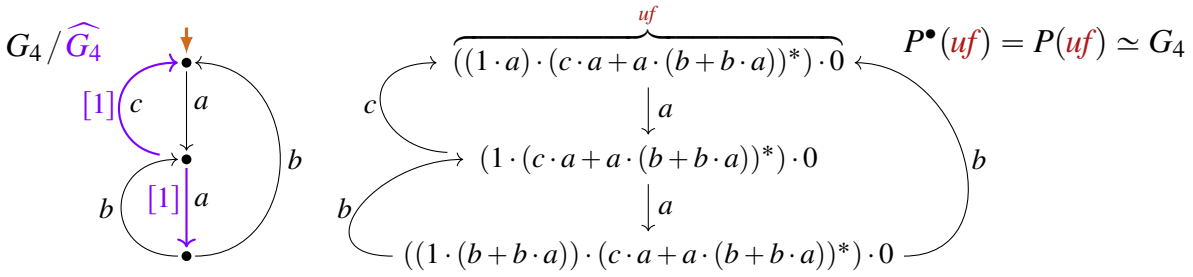

Figure 3: Refined extraction from a LLEE-witness $\widehat{G_4}$ of the graph G_4 in Fig. 1, the bisimulation collapse of graph G_3 in Fig. 1, yields the under-star-1-free regular expression uf, and makes it possible to recognize G_4 as $P^\bullet$-expressible, and also as P-expressible (since G_4 has a single scc, the shortcoming of P described in Sect. 2 is inconsequential here). Note the subtle differences between uf (the use of 1) and the expression $(a\cdot(c\cdot a + a\cdot(b+b\cdot(a+a)))^*)\cdot 0$ (the occurrence of $a+a$ for a) whose process interpretation is G_3.

(I)$^{(+\backslash *)}_{P^\bullet}$: Compact process ***i***nterpretations $P^\bullet(uf)$ of under-star-1-free regular expressions $uf \in RExp^{(+\backslash *)}(A)$ are finite process graphs with actions in A that satisfy LLEE.

3 Refined extraction yields collapse result for the compact interpretation

The crucial tool for obtaining our main result, image-closedness under bisimulation collapse of $P^\bullet$ for under-star-1-free expressions, is a sharpening of the extraction statement **(E)**$^{(+\backslash *)}_{P}$ that holds for the compact process interpretation $P^\bullet$. While **(E)**$^{(+\backslash *)}_{P}$ can be transferred to the compact process interpretation $P^\bullet$ easily, a technical refinement of the extraction procedure is able to yield the following stronger statement:

(E$^+$)$^{(+\backslash *)}_{P^\bullet}$: From every finite process graph G with LLEE and with actions in A an under-star-1-free regular expression $uf \in RExp^{(+\backslash *)}(A)$ with unary star $*$ can be ***e***xtracted such that $G \rightrightarrows P^\bullet(uf)$ holds (that is, there is a functional bisimulation from G to the compact process interpretation $P^\bullet(uf)$ of uf).

The refined extraction procedure from which this statement can be proved is a careful adaptation of the extraction procedures from process graphs with LLEE as described in [11, 6, 9]. While the formulation of this procedure is beyond the scope of this extended abstract, we refer to the recent report [10] where the details are described in Appendix A.1. As for an example we refer to Fig. 3, in which we give the under-star-1-free regular expression uf obtained by refined extraction from the collapsed graph G_4 with LLEE from Fig. 1, and illustrate that the compact process interpretation $P^\bullet(uf)$ of uf is isomorphic to G_4 itself. The concept of 'LLEE-witness' from [11] used in Fig. 3 is explained informally in the appendix.

Now in view of that fact that every functional bisimulation from a bisimulation collapsed process graph is also an isomorphism, **(E$^+$)**$^{(+\backslash *)}_{P^\bullet}$ directly implies the following statement:

(CE$^+$)$^{(+\backslash *)}_{P^\bullet}$: From every finite process graph G with actions in A and with LLEE that is ***c***ollapsed an under-star-1-free regular expression $uf \in RExp^{(+\backslash *)}(A)$ with unary star $*$ can be ***e***xtracted such that $G \simeq P^\bullet(uf)$ holds (that is, G is isomorphic to the compact process interpretation $P^\bullet(uf)$ of uf).

Furthermore the collapse statement **(C)**$_{\mathrm{st}}$ can be extended easily to all process graphs with LLEE:

(C): The class of finite process graphs with LLEE is closed under bisimulation ***c***ollapse.

In this way we obtain our main result, and also an expressibility statement analogous to **(Exp)**$^{(+)}_{P_{\circledast}}$:

(IC)$^{(+\backslash *)}_{P^\bullet}$: The ***i***mage of the ***c***ompact process interpretation $P^\bullet$, when restricted to under-star-1-free star expressions in $RExp^{(+\backslash *)}(A)$ for action set A, is closed under bisimulation collapse, mod. isomorphism.

(Exp)$^{(\pm\backslash *)}_{P,P^\bullet}$: A finite process graph is ***exp***ressible by an under-star-1-free regular expression with unary star $*$ if and only if its bisimulation collapse satisfies LEE (and, by **(IC)**$^{(\pm\backslash *)}_{P^\bullet}$, is in the image of $P^\bullet$).

Hereby **(IC)**$^{(\pm\backslash *)}_{P^\bullet}$ follows directly from **(I)**$^{(\pm\backslash *)}_{P^\bullet}$, **(C)**, and **(CE**$^+$**)**$^{(\pm\backslash *)}_{P^\bullet}$. Finally **(Exp)**$^{(\pm\backslash *)}_{P,P^\bullet}$ follows, for "$\Rightarrow$" from **(I)**$^{(\pm\backslash *)}_{P^\bullet}$, and **(C)**, and for "$\Leftarrow$" from **(CE**$^+$**)**$^{(\pm\backslash *)}_{P^\bullet}$ (or also from **(E**$^+$**)**$^{(\pm\backslash *)}_{P^\bullet}$).

References

[1] Jos Baeten, Flavio Corradini & Clemens Grabmayer (2007): *A Characterization of Regular Expressions Under Bisimulation*. Journal of the ACM 54(2), pp. 1–28, doi:10.1145/1219092.1219094.

[2] Doeko Bosscher (1997): *Grammars Modulo Bisimulation*. Ph.D. thesis, University of Amsterdam.

[3] Irving M. Copi, Calvin C. Elgot & Jesse B. Wright (1958): *Realization of Events by Logical Nets*. Journal of the ACM 5(2), doi:10.1007/978-1-4613-8177-8_1.

[4] Clemens Grabmayer (2020): *Structure-Constrained Process Graphs for the Process Semantics of Regular Expressions*. Technical Report, `arxiv.org`, doi:https://doi.org/10.48550/arXiv.2012.10869. arXiv:2012.10869. Report version of [7].

[5] Clemens Grabmayer (2021): *A Coinductive Version of Milner's Proof System for Regular Expressions Modulo Bisimilarity*. Technical Report, `arxiv.org`, doi:10.48550/arXiv.2108.13104. arXiv:2108.13104. Extended report for [6].

[6] Clemens Grabmayer (2021): *A Coinductive Version of Milner's Proof System for Regular Expressions Modulo Bisimilarity*. In Fabio Gadducci & Alexandra Silva, editors: 9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021), Leibniz International Proceedings in Informatics (LIPIcs) 211, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 16:1–16:23, doi:10.4230/LIPIcs.CALCO.2021.16. Extended report see [5].

[7] Clemens Grabmayer (2021): *Structure-Constrained Process Graphs for the Process Semantics of Regular Expressions*. Electronic Proceedings in Theoretical Computer Science 334, p. 29–45, doi:10.4204/eptcs.334.3. Extended report for [4].

[8] Clemens Grabmayer (2022): *Milner's Proof System for Regular Expressions Modulo Bisimilarity is Complete (Crystallization: Near-Collapsing Process Graph Interpretations of Regular Expressions)*. In: Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '22, Association for Computing Machinery, New York, NY, USA, pp. 1–13.

[9] Clemens Grabmayer (2023): *A Coinductive Reformulation of Milner's Proof System for Regular Expressions Modulo Bisimilarity*. Logical Methods in Computer Science Volume 19, Issue 2, doi:10.46298/lmcs-19(2:17)2023. Available at `https://lmcs.episciences.org/11519`.

[10] Clemens Grabmayer (2023): *The Image of the Process Interpretation of Regular Expressions is Not Closed under Bisimulation Collapse*. Technical Report `arXiv:2303.08553`, `arxiv.org`. Version 2 (November): extension of Prop. 2.12 to 'under star 1-free' expressions, and correction in its proof (added termination subterm to extraction function).

[11] Clemens Grabmayer & Wan Fokkink (2020): *A Complete Proof System for 1-Free Regular Expressions Modulo Bisimilarity*. In: Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '20, Association for Computing Machinery, New York, NY, USA, p. 465–478, doi:10.1145/3373718.3394744.

[12] Stephen C. Kleene (1951): *Representation of Events in Nerve Nets and Finite Automata*. In: Automata Studies, Princeton University Press, Princeton, New Jersey, USA, pp. 3–42, doi:10.1515/9781400882618-002.

[13] Robin Milner (1984): *A Complete Inference System for a Class of Regular Behaviours*. Journal of Computer and System Sciences 28(3), pp. 439–466, doi:10.1016/0022-0000(84)90023-0.