Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Counting nondeterministic computations

Qizhe Yang, Yuxi Fu *

BASICS, Shanghai Jiao Tong University, China

ARTICLE INFO

Article history:
Received 7 June 2020
Received in revised form 8 August 2021
Accepted 17 August 2021
Available online xxxx
Communicated by R. van Glabbeek

Keywords:
Branching bisimulation
Nondeterminism
Divergence

ABSTRACT

The structure of nondeterministic computations is extremely complicated. C-graphs are abstract representations of the branching structure of nondeterministic computations. The paper investigates the structure of finite state nondeterministic computations by showing that the complexity of the structure increases non-elementarily while the number of computation steps increases. This is achieved by establishing a recursive equation relating the number of C-graphs of a certain height to the number of C-graphs of smaller height.

1. Introduction

Computations can be studied in many different models. In computation theory a computation model, say the Turing machine model [27], the λ-calculus model [2], the recursive function model [12], defines a closed system in which programs, machine configurations, closed λ-terms, recursive functions, do not interact. Input and output actions stay at meta level. The equality relation for computation in all these models is the extensional equality $=$ defined as follows: $f = g$ if and only if for every x, either $f(x) = g(x)$ whenever one side is defined, or both $f(x)$ and $g(x)$ are undefined. If one program delivers a result upon an input and another program never stops computing on the input, the two programs are deemed unequal. Evidently divergence plays a central role in the computation theory. It should also play a central role in concurrency theory, which is meant to be an interactive extension of the computation theory [18].

As the foundation of the computation theory Church-Turing Thesis [16] is valid at the operational level [4]. There must be a translation $\widehat{\ }$ from the computational objects of the form M to closed λ-terms. The map $\widehat{\ }$ essentially defines a binary relation $\mathcal{R}$ that satisfies the following property whenever $M \mathcal{R} L$, where M is a machine configuration and L is a closed λ-term: (i) If $M \to M'$ then $L \to^* L'$ such that $M' \mathcal{R} L'$; and (ii) if $L \to L'$ then $M \to^* M'$ such that $M' \mathcal{R} L'$. This is actually the idea of bisimulation [18,23], or more precisely that of barbed bisimulation [20]. The relation $\mathcal{R}$ satisfies an additional property. If $M_0 \mathcal{R} L_0$ and there is an infinite computation sequence $M_0 \to M_1 \to M_2 \to \ldots$, then there is an infinite computation sequence from L_0. It is impossible that $L_0 \to L'$ for some L' such that L' cannot do any computation because a terminating computation is never equal to a divergent computation according to the definition of extensional equality. In other words $\mathcal{R}$ must be divergence sensitive.

The classic Turing machines have been extended to nondeterministic Turing machines for various purposes. The best framework to study nondeterministic computations is offered by concurrency theory. In this theory a model defines the operational semantics of processes whose primary task is to interact with each other [15,18]. Computation is seen as a cooperation between two interacting processes. When interactions are explicit, the idea of observations as tests for equality

* Corresponding author.
E-mail addresses: spacepenguin9494@gmail.com (Q. Yang), fu-yx@cs.sjtu.edu.cn (Y. Fu).

https://doi.org/10.1016/j.tcs.2021.08.022
0304-3975/

becomes natural. Bisimulation then comes out as the finest characterization of the branching structure of nondeterminism caused by interaction, extending the input-output criterion of the extensional equality. It turns out that a minimal formulation of bisimulation gives rise to an equality, the barbed equality, which applies to all concurrent models. Such a model independent theory was initiated by Milner and Sangiorgi [20] and followed by the work carried out in [22,13,8]. A systematic study of the model independent theory of interaction is carried out in [6,7]. With this theory it can be argued that there is a unique equality for both computation and interaction.

There have been many studies on the qualitative aspect of processes [15,14,18,19,25,21]. Algebraic theory of observation forms a considerable portion of process theory. There has been however little research on the quantitative aspect of processes. Given for example an n-state CCS process that may interact at three channels a, b, c, how many unequal processes are there? Questions of this kind are interesting because they help to see the complexity of nondeterminism. In this paper we carry out a quantitative study of nondeterministic computational objects. By computational objects we mean processes that cannot interact with any other processes. We may think of them as processes that pick up some input at the beginning of computation and output a particular value by the end of the computation. The reason that we focus on these objects is twofold. Firstly the simplification allows us to study the branching structure caused solely by the internal actions of processes. External actions would complicate the investigation. Quantitative theory of general processes should begin with a study of the nondeterministic computational objects. Secondly as is pointed out in [5] these objects already demonstrate nice and rich structures that we understood very little before. Methodologies for the study of the nondeterministic computational objects could hopefully be applied to the quantitative investigations of interactive processes.

We consider the finite state computational objects defined by a variant $\mathbb{CCS}^{\mu}$ of CCS that admits only τ actions. Like in [5] we will focus on an abstract representation of the computational objects called C-graphs. In this paper we classify computational objects by the worst case branching time complexity; equivalently we classify C-graphs according to graph height. The main contributions of the paper are as follows:

- A recursive equality for the number L_n of C-graphs of height n is derived. Additionally it is shown that the growth rate of L_n is non-elementary. These are reported in Section 3.
- Similar results are derived for k-regular C-graphs. This is done in Section 4.
- The relationship between the number of C-graphs counted by graph height and the number of C-graphs counted by graph size is characterized by a set of inequalities, giving both upper and lower bounds for one in terms of the other. This is described in Section 5.

A few comments are made in the final section.

2. C-graph

Nondeterminism is an intrinsic phenomenon of concurrency. Nondeterministic computation is best demonstrated in a process model. Consider $A = \mu X.(\tau.\mathbf{0} + \tau.(a \mid X) + \tau.\Omega_a)$ defined in the syntax of process model CCS, where $\Omega_a = \mu Y.\tau.(a \mid Y)$. In a process model a computation step is formalized by a τ transition. Observe that $A \xrightarrow{\tau} \mathbf{0}$, meaning that A may terminate after one step computation. Also $A \xrightarrow{\tau} a \mid A$ and $A \xrightarrow{\tau} \Omega_a$ are one step computations. The process $a \mid A$ is not equivalent to A since in the former the ability to synchronize in channel a is persistent, which is not the case for A. The process Ω_a is not equivalent to A either because Ω_a can synchronize at channel a again and again, whereas A may preempt any synchronization at a from happening. Moreover $a \mid A$ and Ω_a are not equivalent due to the preemptive power of A. Let's slightly modify this example. Define $A_0 = \mu X.(\tau.\mathbf{0} + \tau.X + \tau.\Omega_0)$, where $\Omega_0 = \mu Y.\tau.Y$. The only action this process can do is computation. After one computation step it either evolves to the null process $\mathbf{0}$, or to the divergent process Ω_0, or to itself. One may argue that these three processes are not equivalent to each other from the point of view of resource consuming. The null process $\mathbf{0}$ does not consume any energy. The divergent process Ω_0 always consumes more and more energy. The process A_0 may terminate after consuming some energy. The equality to be introduced in a moment is consistent with this resource consuming viewpoint.

We shall introduce two formalisms for nondeterministic computational objects. Firstly we introduce these objects using the familiar process notation [5]. Consider a variant $\mathbb{CCS}^{\mu}$ of CCS that admits only τ actions. The *finite state terms* are generated from the following BNF:

$$T := X \mid S \mid \Delta(T) \mid \mu X.T,$$
$$S := \mathbf{0} \mid \tau.T \mid S + S.$$

In the above X is a term variable, S is a nondeterministic term, and $\mu X.T$ is a recursive term in which X is bound. The term $\Delta(T)$ either behaves as T or evolves to itself. A *finite state computational object* is a term that does not contain any free term variables. We write P, Q for finite state computational objects. The operational semantics of $\mathbb{CCS}^{\mu}$ is defined by the following rules.

$$\frac{}{\tau.T \xrightarrow{\tau} T} \qquad \frac{S_i \xrightarrow{\tau} S_i' \quad i \in \{0,1\}}{S_0 + S_1 \xrightarrow{\tau} S_i'} \qquad \frac{}{\Delta(T) \xrightarrow{\tau} \Delta(T)} \qquad \frac{T \xrightarrow{\tau} T'}{\Delta(T) \xrightarrow{\tau} T'} \qquad \frac{T\{\mu X.T/X\} \xrightarrow{\tau} T'}{\mu X.T \xrightarrow{\tau} T'}$$

The transition $\Delta(T) \xrightarrow{\tau} \Delta(T)$ is called a *self-loop*. The simplest terminating computational object is $\mathbf{0}$, and the simplest divergent computational object is $\Omega = \Delta(\mathbf{0})$. In the rest of the paper we write $\stackrel{\tau}{\Longrightarrow}$ ($\Longrightarrow$) for the (reflexive and) transitive closure of $\xrightarrow{\tau}$.

We now motivate the equality of the paper. From the point of computation a terminating computation is never equal to a divergent one. An equality on nondeterministic computation objects has to be divergence respectful. There are a number of ways to define divergence respectful relations [28,1,17,11,8,6,10]. We formulate the dichotomy between termination and divergence using an idea of Priese [24]. In what follows we write $\mathcal{R}^{-1} = \{(P, Q) \mid (Q, P) \in \mathcal{R}\}$ for the reverse relation of $\mathcal{R}$.

Definition 1. A binary relation $\mathcal{R}$ is *codivergent* if the following statements are valid:

- If $P_0 \mathcal{R} Q_0 \xrightarrow{\tau} Q_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q_n \xrightarrow{\tau} \ldots$, then $P_0 \stackrel{\tau}{\Longrightarrow} P_1 \mathcal{R} Q_j$ for some P_1 and some $j > 0$.
- If $P_0 \mathcal{R}^{-1} Q_0 \xrightarrow{\tau} Q_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q_n \xrightarrow{\tau} \ldots$, then $P_0 \stackrel{\tau}{\Longrightarrow} P_1 \mathcal{R}^{-1} Q_j$ for some P_1 and some $j > 0$.

The codivergence property has been used to define process equality, see for example [6,10]. The two processes $\mathbf{0}, \Omega$ for example are weakly bisimilar. They are not however related by any codivergent relation. The reason that Definition 1 is the right formulation in our case is that it goes with the notion of bisimulation hand in hand. Bisimulation equality [18,23] is the standard equality for nondeterministic processes that takes into account of the branching structure. We shall use a refinement of the weak bisimulation introduced by van Glabbeek and Weijland [9].

Definition 2. A binary relation $\mathcal{R}$ is a *bisimulation* if the following statements are valid:

- If $P \mathcal{R} Q \xrightarrow{\tau} Q'$ then one of the following statements is valid:
  - $P \Longrightarrow P'$ for some P' such that $P' \mathcal{R} Q$ and $P' \mathcal{R} Q'$.
  - $P \Longrightarrow P'' \mathcal{R} Q$ for some P'' such that $P'' \xrightarrow{\tau} P' \mathcal{R} Q'$ for some P'.
- If $Q \mathcal{R}^{-1} P \xrightarrow{\tau} P'$ then one of the following statements is valid:
  - $Q \Longrightarrow Q'$ for some Q' such that $Q' \mathcal{R}^{-1} P$ and $Q' \mathcal{R}^{-1} P'$.
  - $Q \Longrightarrow Q'' \mathcal{R}^{-1} P$ for some Q'' such that $Q'' \xrightarrow{\tau} Q' \mathcal{R}^{-1} P'$ for some Q'.

By the standard approach in concurrency theory we can define the equality between the nondeterministic computational objects as follows: P and Q are *equal*, notation $P \simeq Q$, if (P, Q) is in a codivergent bisimulation.

There are infinitely many finite state computational objects. In [5] the author defined $\Upsilon_0 = \mathbf{0}$, $\Upsilon_1 = \Delta(\tau.\mathbf{0})$, $\Upsilon_2 = \tau.\mathbf{0} + \tau.\Omega$, and introduced the inductive definition

$$\Upsilon_{2i+1} = \Delta(\tau.\mathbf{0} + \tau.\Upsilon_{2i}),$$
$$\Upsilon_{2i+2} = \tau.\mathbf{0} + \tau.\Omega + \tau.\Upsilon_{2i+1}.$$

It is clear that Υ_1 diverges whereas Υ_0 does not, and Υ_2 may terminate but Υ_1 cannot. To help the reader get a glimpse of the complexity of C-graphs we repeat the argument that the infinite sequence $\Upsilon_0, \Upsilon_1, \Upsilon_2, \ldots$ are pairwise unequal. Suppose $\forall j < 2i-1.\ \Upsilon_j \neq \Upsilon_{2i-1}$ and $\forall j < 2i.\ \Upsilon_j \neq \Upsilon_{2i}$. For any even number k smaller than $2i+1$, $\Upsilon_k \neq \Upsilon_{2i+1}$ because Υ_{2i+1} loops while Υ_k does not, and $\Upsilon_k \neq \Upsilon_{2i+2}$ by induction. If k is an odd number smaller than $2i+1$, then $\Upsilon_k \neq \Upsilon_{2i+1}$ by induction, and $\Upsilon_k \neq \Upsilon_{2i+2}$ because $\Upsilon_{2i+2} \xrightarrow{\tau} \Omega$ cannot be matched by any computation sequence from Υ_k. For more examples of computational objects consult [5].

It is worth remarking that the so-called weak bisimilarity [18] cannot be used to study the branching structure of computational objects. It would identify all the finite state computational objects definable in $\mathbb{CCS}^{\mu}$.

A one step computation $P \xrightarrow{\tau} P'$ can be depicted as a directed edge with the two nodes labeled by P and P' respectively. A one step computation $P \xrightarrow{\tau} P$ is then a self-loop with the node labeled by P. The set of all computation paths from P form a rooted graph. If we remove the labels from the nodes of the graph, we get a graph that is an abstract representation of a nondeterministic computation structure. It is called a D-graph in [5].

Definition 3. A directed graph $G = (V, E)$ is a *D-graph* if the following hold:

- There is precisely one node, the root, that does not have any incoming edge.
- There is at most one edge from a node to another. There is at most one edge from a node to itself.
- Every node is reachable from the root.

A *self-loop* is an edge from a node to itself. The *out-degree* of a node is the number of out-going edges of the node, *including* the self-loop. If there is an edge from a node to a *different* node, the latter is a *child* of the former. If there is a path from a node to a different node, the latter is a *descendant* of the former. For ease of reference we shall often label

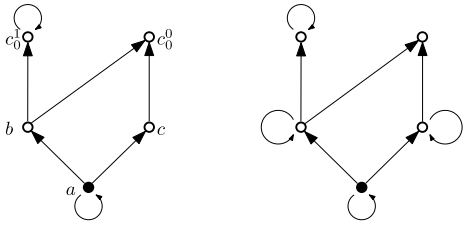

Fig. 1. The left is a D-Graph but not a C-graph. The right is a C-graph.

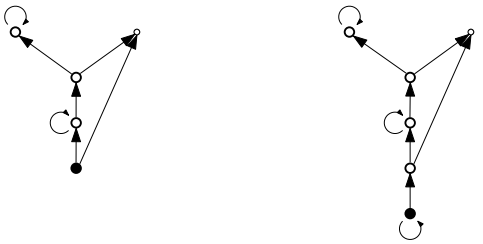

Fig. 2. Height Versus Size of a C-Graph.

a node of a D-graph by a small case letter. We label the trivial single node D-graph by c_0^0 and the trivial single self-loop node by c_0^1. Evidently every D-graph contains at least one of the two trivial D-graphs as subgraph. By D being an (induced) subgraph of D' we mean that whenever the two nodes of an edge of D' are in D then the edge is in D. An *internal node* has at least one out-going edge to a different node.

We denote a D-graph by $\mathcal{G} = (G, c)$, where G is a directed graph and c is the root. A D-graph is a syntax free representation of a finite state computational object. Consider the two D-graphs presented in Fig. 1, in which the bullets are the roots. The left is an abstract representation of the operational behavior of $\Delta(\tau.\tau + \tau.(\tau.\Omega + \tau))$ and the right an abstract representation of $\Delta(\tau.\Delta(\tau) + \tau.\Delta(\Omega + \tau))$. For ease of reference the nodes of the left are labeled. Every node in the D-graph represents a computational object. The computational object $\Delta(\tau.\tau + \tau.(\tau.\Omega + \tau))$ is represented by the root labeled a. The object $\tau.\Omega + \tau$ is represented by b. The one step computation $\Delta(\tau.\tau + \tau.(\tau.\Omega + \tau)) \xrightarrow{\tau} \tau.\Omega + \tau$ is represented by the edge from a to b. It is clear that the equality $\simeq$ can be applied to the nodes of D-graphs. It equates the nodes labeled c_0^0 and c because the relation $\{(a,a), (b,b), (c_0^1, c_0^1), (c_0^0, c_0^0), (c,c), (c, c_0^0), (c_0^0, c)\}$ is a codivergent bisimulation. In an abstract representation of computational object there is no point in introducing equal nodes. Hence the following.

Definition 4. A *C-graph* $\mathcal{G} = (G, c)$ is a D-graph in which no two nodes are equal.

The right D-graph in Fig. 1 is a C-graph. When no confusion may arise a C-graph will be referred to by the label of its root.

Consider another example. Two C-graphs are defined in Fig. 2. In the left one the root is not equal to the node above the root. This is because the former can reach c_0^0 whereas the latter cannot reach c_0^0 without passing an unequal node. In the right diagram the root is not equal to the node above it either because the former can loop while the latter cannot. It is easy to see from this example how to construct a C-graph of n nodes whose height is $n - 2$. The notion of C-graph height will be formalized in the next section.

We will call the single node C-graphs c_0^0 and c_0^1 the trivial C-graphs. Almost all C-graphs contain both c_0^0 and c_0^1 as leaves. It is easy to verify that if a C-graph does not contain c_0^0, it is nothing but c_0^1. If a C-graph does not contain c_0^1, it is either c_0^0 or the two node C-graph whose root can loop.

3. Counting C-graphs

From the complexity theoretical viewpoint we are mostly interested in the maximal number of steps a computational object may engage. C-graphs are computational objects. It is therefore natural to classify them by the length of the longest admitted paths. Here self-loops are ignored when counting the length of a path. So we are counting the number of computational steps that change states. In other words C-graphs are classified by the worst case branching time complexity. Suppose u, v are distinct nodes in a C-graph $\mathcal{G}$. The *distance* $dist(u, v)$ is the length of the longest path from u to v that contains no self-loops. If there is no path from u to v, let $dist(u, v)$ be for instance -1. We define the *height* $\mathfrak{h}(u)$ of u as follows:

$$\mathfrak{h}(u) = \max\left\{dist\left(u, c_0^0\right), dist\left(u, c_0^1\right)\right\}.$$

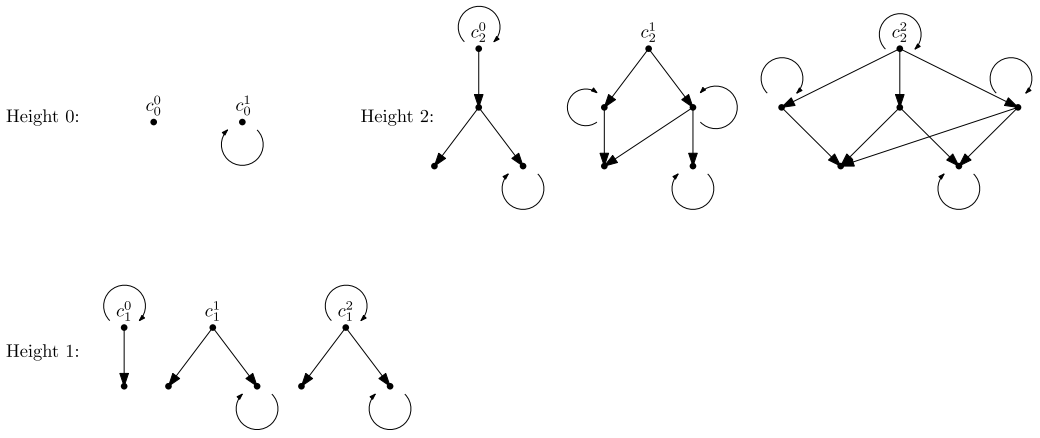

Fig. 3. Examples of C-graph Classified by Height.

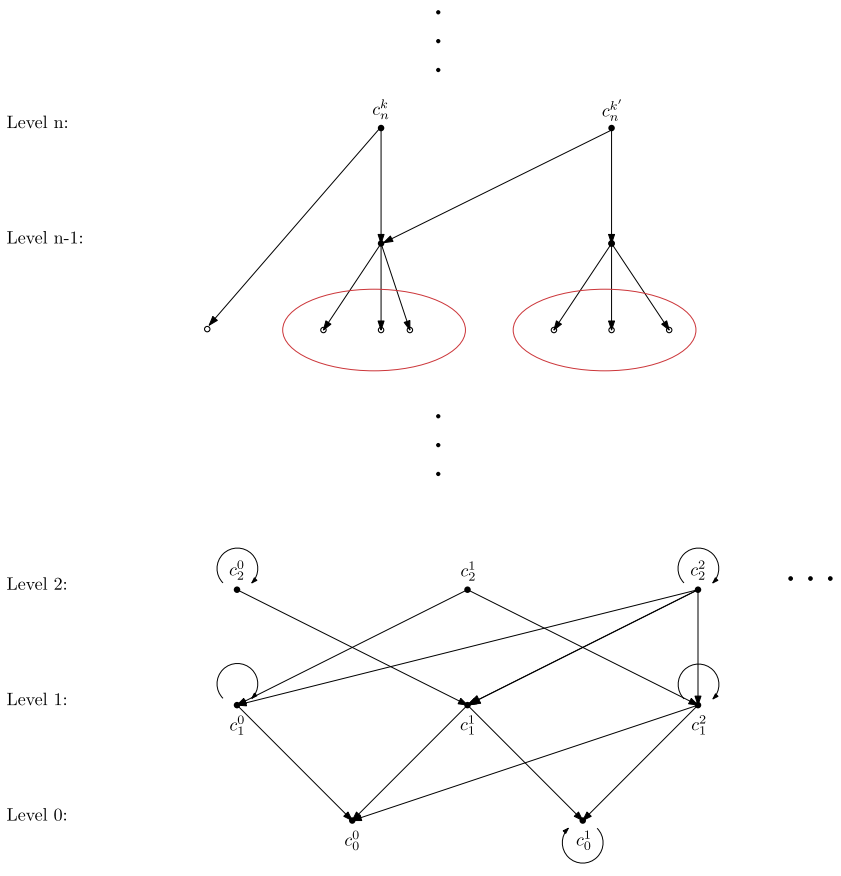

Fig. 4. Finite C-Graphs Arranged in Levels.

We define the *height* $\mathfrak{h}(\mathcal{G})$ of the C-graph $\mathcal{G}$ by the height of the root. The height of c_0^0 and c_0^1 is 0. Here the subscript 0 indicates the height, and the superscripts $0, 1$ are used to differentiate the two nodes. There are three C-graphs of height 1, whose roots are denoted respectively by c_1^0, c_1^1 and c_1^2, see Fig. 3, where three C-graphs of height 2, rooted respectively at c_2^0, c_2^1, c_2^2, are also given.

A C-graph is of height $i+1$ because the children of its root are of height no more than i and at least one child is of height i. This observation allows us to visualize the class of the finite C-graphs as forming a connected infinite directed graph without any root. The layout of the infinite graph is such that the root of a graph of height i stays at the i-th level. The picture is given in Fig. 4. The nodes c_0^0, c_0^1 of Fig. 3 are at the 0-th level in the infinite C-graph of Fig. 4. The roots c_1^0, c_1^1 and c_1^2 are at the first level, and the roots c_2^0, c_2^1, c_2^2 are at the second level. Inductively a node, say c_n^k, at the n-th level

of the infinite graph is the root of a finite C-graph of height n. This finite C-graph consists of all the nodes reachable from c_n^k. In this way we can talk about the number of the finite C-graphs at the n-th level. We further clarify the picture in Fig. 4 by the following observations.

- Nodes at the same level are disconnected. If a and b are distinct nodes at the n-th level and there is an edge from a to b, then either one of the nodes is in the wrong level or the graph rooted at a is not a C-graph because $a \simeq b$.
- Apart from self-loops, all out-going edges of a node a at the n-th level point to nodes at lower levels. There is at least one out-going edge that points to a node at the $(n-1)$-th level, otherwise a would not be in the n-th level.
- Let $C(a)$ be the set of the children of a node a at the n-th level. Let $D(a)$ be 1 if a has a self-loop and be 0 otherwise. Suppose that b is a node at the n-th level that is distinct from a. Then either $C(a) \neq C(b)$ or $D(a) \neq D(b)$. This is because if $C(a) = C(b)$ and $D(a) = D(b)$, then $\simeq \cup \{(a, b)\}$ is a codivergent bisimulation, and consequently $a \simeq b$, contradicting to the assumption.
- Suppose a is a node at the n-th level and a' is a node at the $(n-1)$-th level. Then either $C(a) \nsubseteq C(a') \cup \{a'\}$ or $D(a) = 1$ and $D(a') = 0$. Otherwise $\simeq \cup \{(a, a'), (a', a)\}$ would be a codivergent bisimulation.

In the n-th level the nodes without self-loops can be classified into two groups. The node c_n^k in Fig. 4 is connected to exactly one node, say c_{n-1}^j, in the $(n-1)$-th level. In this case c_n^k must connect to at least one node below the $(n-1)$-th level that is not a child of c_{n-1}^j. The node $c_n^{k'}$ in Fig. 4 is connected to at least two nodes in the $(n-1)$-th level. It may or may not connect to any node in lower levels.

Before counting the number of C-graphs classified by levels, let's introduce some notations. We write $\mathcal{L}_n$ for the set of all C-graphs in the n-th level, and $L_n = |\mathcal{L}_n|$ for the size of $\mathcal{L}_n$. To find out the relationship between L_n and L_{n-1} we need to look at subclasses of $\mathcal{L}_n$ induced by the degree of roots. The notation $\mathcal{L}_{n,\circlearrowleft}^j$ stands for the subset of $\mathcal{L}_n$ containing those C-graphs whose roots have self-loop and have out-degree j. Similarly $\mathcal{L}_{n,\bullet}^j$ denotes the subset of $\mathcal{L}_n$ containing root without self-loop and with out-degree j. Let $L_{n,\circlearrowleft}^j = |\mathcal{L}_{n,\circlearrowleft}^j|$ and $L_{n,\bullet}^j = |\mathcal{L}_{n,\bullet}^j|$. The set of all the C-graphs that stay at or below the n-th level is $\mathcal{S}_n = \bigcup_{i=0}^n \mathcal{L}_i$. Let $S_n = |\mathcal{S}_n|$. The following equalities hold by definition.

$$\begin{aligned} L_n^j &= L_{n,\bullet}^j + L_{n,\circlearrowleft}^j, \\ L_n &= \sum_{j=1}^{S_{n-1}+1} L_n^j. \end{aligned} \tag{1}$$

In (1) we have taken into account the self loop of the root. Notice that $L_{n,\circlearrowleft}^{S_{n-1}+1} = 1$. We will derive an equality for $L_{n,\bullet}^j$ that guarantees $L_{n,\bullet}^{S_{n-1}+1} = 0$ and $L_{n,\bullet}^{S_{n-1}} = 1$.

The main result of this section is stated next.

Theorem 3.1. *The following recursive equality holds for $n \geq 3$.*

$$S_n = \left(\sum_{k=2}^{n} (-1)^k \frac{k!}{k-1} \cdot k^{S_{n-(k-1)}} \right) - (-1)^n (n-1)! \cdot (n^3 + 2n^2 + n + 1).$$

The theorem enables one to calculate S_n in an inductive manner and to compute L_n by the equality $L_n = S_n - S_{n-1}$. We will prove Theorem 3.1 in Section 3.1 through Section 3.4.

3.1. Node with a single next level child

The standard notation for the number of combinations of choosing k items from a total of n items is $\binom{n}{k}$. We shall allow n to be smaller than k. Thus

$$\binom{n}{k} = \begin{cases} \frac{n!}{(n-k)!\cdot k!}, & \text{if } n \geq k, \\ 0, & \text{if } n < k. \end{cases}$$

Recall that $0! = 1$. To simplify the combinatorial argument, we even allow n to take the negative value -1. The following combinatorial equations will be useful in present paper.

$$\binom{K}{k-1} + \binom{K}{k} = \binom{K+1}{k}, \tag{2}$$

$$\sum_{i=0}^{k} \binom{K-K'}{i} \binom{K'}{k-i} = \binom{K}{k}, \tag{3}$$

$$\sum_{i=0}^{k} d^i \binom{k}{i} = (d+1)^k. \tag{4}$$

Given a C-graph c_{n-1} in the $(n-1)$-th level, we construct a C-graph at the n-th level by introducing a new root c_n, divergent or not, and connecting it to c_{n-1}. We assume that the new node does not connect to any other node in the $(n-1)$-th level. Unless c_n is divergent and c_{n-1} is not divergent, the new root must connect to at least one node in $\mathcal{S}_{n-2}$ for otherwise the new root c_n would be equal to the old root c_{n-1}. Let $\mathcal{A}^j_{n,\circlearrowleft}$ be the subset of $\mathcal{L}^j_{n,\circlearrowleft}$ containing all the C-graphs with precisely one child in the $(n-1)$-th level, and let $A^j_{n,\circlearrowleft} = |\mathcal{A}^j_{n,\circlearrowleft}|$. Similarly we define $\mathcal{A}^j_{n,\bullet}$ and $A^j_{n,\bullet}$. The next lemma explains how to calculate $A^j_{n,\circlearrowleft}$ and $A^j_{n,\bullet}$.

Lemma 3.2. *The following equalities are valid.*

1. $A^j_{n,\circlearrowleft} = \sum_{i=1}^{S_{n-2}+1} \left(L^i_{n-1,\bullet} \cdot \binom{S_{n-2}}{j-2} + \sum_{t=1}^{j-2} L^i_{n-1,\circlearrowleft} \cdot \binom{S_{n-2}-i+1}{t} \cdot \binom{i-1}{j-2-t} \right)$.
2. $A^j_{n,\bullet} = \sum_{i=1}^{S_{n-2}+1} \sum_{t=1}^{j-1} \left(L^i_{n-1,\bullet} \cdot \binom{S_{n-2}-i}{t} \cdot \binom{i}{j-1-t} + L^i_{n-1,\circlearrowleft} \cdot \binom{S_{n-2}-i+1}{t} \cdot \binom{i-1}{j-1-t} \right)$.

Proof. We prove the first equality. The proof for the second one is similar. Fix j, i. There are two cases.

- The root c_n of a C-graph in $\mathcal{A}^j_{n,\circlearrowleft}$ connects to a root c_{n-1} in $\mathcal{L}^i_{n-1,\bullet}$. Since c_n has a self-loop and c_{n-1} does not, c_n is already not equal to c_{n-1}. This means that we can choose the rest of the $j-2$ nodes from $\mathcal{S}_{n-2}$ in any way we want. This explains the summand $L^i_{n-1,\bullet} \cdot \binom{S_{n-2}}{j-2}$.
- The root c_n of a C-graph in $\mathcal{A}^j_{n,\circlearrowleft}$ connects to a root c_{n-1} in $\mathcal{L}^i_{n-1,\circlearrowleft}$. The root c_n should connect to at least one root in $\mathcal{S}_{n-2}$; otherwise c_n would be equal to c_{n-1}. For $t \in [j-2]$ the root c_n may connect to any $j-2-t$ nodes among the $i-1$ children of c_{n-1}, disregarding the self-loop, and to any t nodes among the other $S_{n-2}-i+1$ nodes in $\mathcal{S}_{n-2}$. This explains the summand $\sum_{t=1}^{j-2} L^i_{n-1,\circlearrowleft} \binom{S_{n-2}-i+1}{t} \binom{i-1}{j-2-t}$.

We remark that $\binom{S_{n-2}-i}{t}$ for example is equal to 0 when $S_{n-2} - i \in \{-1, 0, \ldots, t-1\}$. □

3.2. Node with more than one next level child

Next we consider the situation in which the root c_n of a C-graph at the n-th level is connected to more than one node in the $(n-1)$-th level. The notations $\mathcal{B}^j_{n,\circlearrowleft}$, $\mathcal{B}^j_{n,\bullet}$, $B^j_{n,\circlearrowleft}$ and $B^j_{n,\bullet}$ are defined in similar fashion. A node that connects to two distinct nodes in $\mathcal{S}_{n-1}$ must stay at the n-th level. If c_n connects to t nodes in the $(n-1)$-th level, it may connect to any $j-t-1$ or $j-t$ nodes in $\mathcal{S}_{n-2}$, depending on if c_n has a self-loop. Hence the following.

$$B^j_{n,\circlearrowleft} = \sum_{t=2}^{j-1} \binom{L_{n-1}}{t} \binom{S_{n-2}}{j-t-1} \text{ and } B^j_{n,\bullet} = \sum_{t=2}^{j} \binom{L_{n-1}}{t} \binom{S_{n-2}}{j-t}. \tag{5}$$

3.3. The proof of Theorem 3.1

We now calculate L^j_n using Lemma 3.2 and the equalities in (5).

$$A^j_{n,\circlearrowleft} + A^j_{n,\bullet} = \sum_{i=1}^{S_{n-2}+1} \left(L^i_{n-1,\bullet} \cdot \binom{S_{n-2}}{j-2} + \sum_{t=1}^{j-2} L^i_{n-1,\circlearrowleft} \cdot \binom{S_{n-2}-i+1}{t} \cdot \binom{i-1}{j-2-t} \right) +$$
$$\sum_{i=1}^{S_{n-2}+1} \sum_{t=1}^{j-1} \left(L^i_{n-1,\bullet} \cdot \binom{S_{n-2}-i}{t} \cdot \binom{i}{j-1-t} + L^i_{n-1,\circlearrowleft} \cdot \binom{S_{n-2}-i+1}{t} \cdot \binom{i-1}{j-1-t} \right).$$

Fix i. In the above expression the coefficient of $L^i_{n-1,\bullet}$ is

$$\binom{S_{n-2}}{j-2} + \sum_{t=1}^{j-1} \binom{S_{n-2}-i}{t} \binom{i}{j-1-t} \overset{(3)}{=} \binom{S_{n-2}}{j-2} + \binom{S_{n-2}}{j-1} - \binom{i}{j-1} \overset{(2)}{=} \binom{S_{n-2}+1}{j-1} - \binom{i}{j-1},$$

and the coefficient of $L^i_{n-1,\circlearrowleft}$ is

$$\begin{aligned}
&\sum_{t=1}^{j-2}\binom{S_{n-2}-i+1}{t}\binom{i-1}{j-2-t}+\sum_{t=1}^{j-1}\binom{S_{n-2}-i+1}{t}\binom{i-1}{j-1-t}\\
&\overset{(3)}{=}\binom{S_{n-2}}{j-2}-\binom{i-1}{j-2}+\binom{S_{n-2}}{j-1}-\binom{i-1}{j-1}\\
&\overset{(2)}{=}\binom{S_{n-2}+1}{j-1}-\binom{i}{j-1}.
\end{aligned}$$

The two coefficients are the same. Thus

$$A^j_{n,\circlearrowleft}+A^j_{n,\bullet}=\sum_{i=1}^{S_{n-2}+1}L^i_{n-1}\cdot\left(\binom{S_{n-2}+1}{j-1}-\binom{i}{j-1}\right)=L_{n-1}\cdot\binom{S_{n-2}+1}{j-1}-\sum_{i=1}^{S_{n-2}+1}L^i_{n-1}\cdot\binom{i}{j-1}.$$

The above equality can be justified by a combinatorial argument. The number $A^j_{n,\circlearrowleft}+A^j_{n,\bullet}$ can be calculated in the following manner: Fix a new root with i children. For each $(n-1)$-th level child, there are $\binom{S_{n-2}}{j-2}+\binom{S_{n-2}}{j-1}=\binom{S_{n-2}+1}{j-1}$ different D-graphs. Thus there are $L_{n-1}\cdot\binom{S_{n-2}+1}{j-1}$ D-graphs altogether. Let's calculate the number of those D-graphs that are not C-graphs. For that purpose we only have to look at the situations where the root is actually equal to the $(n-1)$-th level node. There are two cases.

- Suppose the $(n-1)$-th level node does not have a self-loop and its out degree is i. The new root is equal to the $(n-1)$-th level node if and only if the other $j-1$ children of the new root are also the children of the $(n-1)$-th level node. There are $\sum_{i=1}^{S_{n-2}+1}L^i_{n-1,\bullet}\cdot\binom{i}{j-1}$ such D-graphs.
- Suppose the $(n-1)$-th level node has a self-loop and its out degree is i. The new root is equal to the $(n-1)$-th level node if and only if either of the following happens: (i) The new root has a self-loop and the $j-2$ children are also the children of the $(n-1)$-th level node. There are $\sum_{i=1}^{S_{n-2}+1}L^i_{n-1,\circlearrowleft}\cdot\binom{i-1}{j-2}$ such D-graphs. (ii) The new root does not have a self-loop and the $j-1$ children of the new root are also the children of the $(n-1)$-th level node. There are $\sum_{i=1}^{S_{n-2}+1}L^i_{n-1,\circlearrowleft}\cdot\binom{i-1}{j-1}$ such D-graphs. Altogether there are $\sum_{i=1}^{S_{n-2}+1}L^i_{n-1,\circlearrowleft}\cdot\binom{i}{j-1}$ such D-graphs.

By summing up the above two expressions one gets $\sum_{i=1}^{S_{n-2}+1}L^i_{n-1}\cdot\binom{i}{j-1}$. We have effectively given two proofs for the equality $A^j_{n,\circlearrowleft}+A^j_{n,\bullet}=L_{n-1}\cdot\binom{S_{n-2}+1}{j-1}-\sum_{i=1}^{S_{n-2}+1}L^i_{n-1}\cdot\binom{i}{j-1}$.

Next we count the number of the C-graphs of height n with more than two $(n-1)$-th level nodes. It follows from (5) and the equality $L_{n-1}=S_{n-1}-S_{n-2}$ that

$$\begin{aligned}
B^j_{n,\circlearrowleft}+B^j_{n,\bullet}&=\sum_{t=2}^{j-1}\binom{L_{n-1}}{t}\binom{S_{n-2}}{j-t-1}+\sum_{t=2}^{j}\binom{L_{n-1}}{t}\binom{S_{n-2}}{j-t}\\
&\overset{(3)}{=}\binom{S_{n-1}}{j-1}-L_{n-1}\binom{S_{n-2}}{j-2}-\binom{S_{n-2}}{j-1}+\binom{S_{n-1}}{j}-L_{n-1}\binom{S_{n-2}}{j-1}-\binom{S_{n-2}}{j}\\
&\overset{(2)}{=}\binom{S_{n-1}+1}{j}-\binom{S_{n-2}+1}{j}-L_{n-1}\cdot\binom{S_{n-2}+1}{j-1}.
\end{aligned}\tag{6}$$

The expression in (6) can also be argued in a combinatorial fashion. To construct a C-graph of height n in $\mathcal{B}^j_n$, we let the n-th level root connect to any j nodes in lower levels or add a self-loop and connect to any $j-1$ nodes in lower levels. There are $\binom{S_{n-1}}{j-1}+\binom{S_{n-1}}{j}=\binom{S_{n-1}+1}{j}$ different constructions. These constructions do not necessarily produce C-graphs in $\mathcal{B}^j_n$. We must remove those that are not in $\mathcal{B}^j_n$. There are two bad cases.

- If the j neighbors of the root are either in $\mathcal{S}_{n-2}$ or is the root itself, the graph is not in $\mathcal{B}^j_n$. There are $\binom{S_{n-2}+1}{j}$ such C-graphs.
- The graph that connects to only one next level child is not in $\mathcal{B}^j_n$. There are $L_{n-1}\cdot\binom{S_{n-2}+1}{j-1}$ such C-graphs.

This completes the combinatorial argument. We can now calculate L^j_n.

$$\begin{aligned}
L^j_n&=L^j_{n,\circlearrowleft}+L^j_{n,\bullet}\\
&=A^j_{n,\circlearrowleft}+B^j_{n,\circlearrowleft}+A^j_{n,\bullet}+B^j_{n,\bullet}
\end{aligned}$$

$$= \binom{S_{n-1}+1}{j} - \binom{S_{n-2}+1}{j} - \sum_{i=1}^{S_{n-2}+1} L_{n-1}^{i} \cdot \binom{i}{j-1}. \tag{7}$$

We calculate L_n by summing up over j and simplify the expression by making use of (4).

$$\begin{aligned} L_n &= \sum_{j=1}^{S_{n-1}+1} L_n^j \qquad (8)\\ &= \sum_{j=1}^{S_{n-1}+1} \left(\binom{S_{n-1}+1}{j} - \binom{S_{n-2}+1}{j} - \sum_{i=1}^{S_{n-2}+1} L_{n-1}^{i} \cdot \binom{i}{j-1} \right) \\ &= \sum_{j=0}^{S_{n-1}+1} \binom{S_{n-1}+1}{j} - \sum_{j=0}^{S_{n-2}+1} \binom{S_{n-2}+1}{j} - \sum_{j=1}^{S_{n-2}+1} \sum_{i=1}^{S_{n-2}+1} L_{n-1}^{i} \cdot \binom{i}{j-1} \\ &\overset{(4)}{=} 2^{S_{n-1}+1} - 2^{S_{n-2}+1} - \sum_{i=1}^{S_{n-2}+1} \left(L_{n-1}^{i} \cdot \sum_{j=1}^{S_{n-2}+1} \binom{i}{j-1} \right) \\ &\overset{(4)}{=} 2^{S_{n-1}+1} - 2^{S_{n-2}+1} - \sum_{i=1}^{S_{n-2}+1} 2^i L_{n-1}^{i}. \qquad (9) \end{aligned}$$

The right hand side of (8), which is equal to $\sum_{j=1}^{S_{n-1}+1} 1^j \cdot L_n^j$, is unfolded to the expression $\sum_{i=1}^{S_{n-2}+1} 2^i L_{n-1}^i$ in (9). The alert reader might wonder if the unfolding can be done inductively. This is indeed the case.

$$\begin{aligned} \sum_{i=1}^{S_{m-1}+1} k^i L_m^i &\overset{(7)}{=} \sum_{i=1}^{S_{m-1}+1} k^i \left(\binom{S_{m-1}+1}{i} - \binom{S_{m-2}+1}{i} - \sum_{t=1}^{S_{m-2}+1} L_{m-1}^{t} \binom{t}{i-1} \right) \\ &= \sum_{i=0}^{S_{m-1}+1} k^i \binom{S_{m-1}+1}{i} - \sum_{i=0}^{S_{m-2}+1} k^i \binom{S_{m-2}+1}{i} - \sum_{i=1}^{S_{m-1}+1} k^i \sum_{t=1}^{S_{m-2}+1} L_{m-1}^{t} \binom{t}{i-1} \\ &\overset{(4)}{=} (k+1)^{S_{m-1}+1} - (k+1)^{S_{m-2}+1} - k \sum_{t=1}^{S_{m-2}+1} L_{m-1}^{t} \sum_{i=1}^{S_{m-1}+1} k^{i-1} \binom{t}{i-1} \\ &= (k+1)^{S_{m-1}+1} - (k+1)^{S_{m-2}+1} - k \sum_{t=1}^{S_{m-2}+1} L_{m-1}^{t} \sum_{i=0}^{t} k^i \binom{t}{i} \\ &\overset{(4)}{=} (k+1)^{S_{m-1}+1} - (k+1)^{S_{m-2}+1} - k \sum_{t=1}^{S_{m-2}+1} L_{m-1}^{t} (k+1)^t. \end{aligned}$$

The equality can be manipulated into the following form:

$$(k-1)! \sum_{i=1}^{S_{m-1}+1} k^i L_m^i = \frac{(k+1)!}{k} ((k+1)^{S_{m-1}} - (k+1)^{S_{m-2}}) - k! \sum_{t=1}^{S_{m-2}+1} (k+1)^t L_{m-1}^t. \tag{10}$$

It is clear from Fig. 3 that $L_0 = 2$, $L_1 = 3$, $L_1^1 = 0$, $L_1^2 = 2$ and $L_1^3 = 1$. By repeatedly applying (10) we get the following sequence of equality.

$$\begin{aligned} L_n &= 2^{S_{n-1}+1} - 2^{S_{n-2}+1} - \sum_{j=1}^{S_{n-2}+1} 2^j L_{n-1}^j. \qquad (11)\\ &= 2^{S_{n-1}+1} - 2^{S_{n-2}+1} - \left(3^{S_{n-2}+1} - 3^{S_{n-3}+1} - 2 \sum_{j=1}^{S_{n-3}+1} 3^j L_{n-2}^j \right) \\ &= \cdots \\ &= \frac{2!}{1}(2^{S_{n-1}} - 2^{S_{n-2}}) - \frac{3!}{2}(3^{S_{n-2}} - 3^{S_{n-3}}) + \cdots + (-1)^n (n-2)! \left(n^{S_1+1} - n^{S_0+1} - (n-1) \sum_{t=1}^{S_0+1} n^t L_1^t \right) \end{aligned}$$

$$= \sum_{k=2}^{n}\left((-1)^k\frac{k!}{k-1}(k^{S_{n+1-k}}-k^{S_{n-k}})\right)+(-1)^{n+1}(n-1)!\sum_{t=1}^{3}n^tL_1^t$$
$$= \sum_{k=2}^{n}\left((-1)^k\frac{k!}{k-1}(k^{S_{n+1-k}}-k^{S_{n-k}})\right)-(-1)^n n!(n^2+2n).$$

We are now in a position to calculate S_n.

$$S_n=\sum_{i=0}^{n}L_i=L_0+L_1+\sum_{i=2}^{n}L_i=5+\sum_{i=2}^{n}\left(\sum_{k=2}^{i}(-1)^k\frac{k!}{k-1}(k^{S_{i+1-k}}-k^{S_{i-k}})+(-1)^{i+1}(i-1)!(2i^2+i^3)\right).$$

Notice that

$$\sum_{i=2}^{n}\sum_{k=2}^{i}(-1)^k\frac{k!}{k-1}(k^{S_{i+1-k}}-k^{S_{i-k}})=\sum_{k=2}^{n}(-1)^k\frac{k!}{k-1}\sum_{i=k}^{n}(k^{S_{i+1-k}}-k^{S_{i-k}})=\sum_{k=2}^{n}(-1)^k\frac{k!}{k-1}(k^{S_{n+1-k}}-k^{S_0}).$$

Therefore

$$\begin{aligned}S_n&=5+\sum_{k=2}^{n}(-1)^k\frac{k!}{k-1}(k^{S_{n+1-k}}-k^{S_0})+\sum_{i=2}^{n}(-1)^{i+1}(i-1)!(2i^2+i^3)\\&=5+\sum_{k=2}^{n}(-1)^k\frac{k!}{k-1}k^{S_{n+1-k}}-\sum_{k=2}^{n}(-1)^k\frac{k!}{k-1}k^2+\sum_{k=2}^{n}(-1)^{k+1}(k-1)!(2k^2+k^3)\\&=5+\sum_{k=2}^{n}(-1)^k\frac{k!}{k-1}k^{S_{n+1-k}}+\sum_{k=2}^{n}\left((-1)^{k+1}(k-1)!(2k^2+k^3)-(-1)^k\frac{k!}{k-1}k^2\right)\\&=\sum_{k=2}^{n}(-1)^k\frac{k!}{k-1}\cdot k^{S_{n-(k-1)}}+\left(\sum_{k=2}^{n}(-1)^{k-1}(k-2)!(k^4+2k^3-2k^2)\right)+5.\end{aligned}\tag{12}$$

We still need to remove the second summation operator in (12).

3.4. Closed formula for $\sum_{k=2}^{n}(-1)^{k-1}(k-2)!(k^4+2k^3-2k^2)$

Let the *alternating factorial* function af be defined by

$$\mathsf{af}(n)=\sum_{k=1}^{n}(-1)^{n-k}k!.$$

There is a closed formula for $\mathsf{af}(n)$. It turns out however that we only need the following recursive equality.

$$\mathsf{af}(n)=n!-\mathsf{af}(n-1).\tag{13}$$

Let A stand for the summation $\sum_{k=2}^{n}(-1)^{k-1}(k-2)!(k^4+2k^3-2k^2)$. We can rewrite A with the help of the alternating factorial function.

$$\begin{aligned}A&=\sum_{k=2}^{n}(-1)^{k-1}(k+2)!-\sum_{k=2}^{n}(-1)^{k-1}k!+\sum_{k=2}^{n}(-1)^{k-1}(k-1)!+\sum_{k=2}^{n}(-1)^{k-1}(k-2)!\\&=\sum_{k=1}^{n}(-1)^{k-1}(k+2)!+\sum_{k=1}^{n}(-1)^{k}k!+\sum_{k=1}^{n-1}(-1)^{k}k!-\sum_{k=1}^{n-2}(-1)^{k}k!-6\\&=(-1)^{n+1}\sum_{k=1}^{n+2}(-1)^{n+2-k}k!+(-1)^n\sum_{k=1}^{n}(-1)^{n-k}k!+(-1)^{n-1}(n-1)!-5\\&=(-1)^{n+1}\left(\mathsf{af}(n+2)-\mathsf{af}(n)+(n-1)!\right)-5\\&\overset{(13)}{=}(-1)^{n+1}\left((n+2)!-(n+1)!+(n-1)!\right)-5\\&=(-1)^{n+1}(n-1)!(n^3+2n^2+n+1)-5.\end{aligned}$$

The proof of Theorem 3.1 is completed.

3.5. Growth rate of L_n

It should now become clear that L_n is bounded by the number of subsets of $\mathcal{L}_n$ of size at least 2. In other words $L_n \geq 2^{L_{n-1}} - L_{n-1} - 1$. The inequality immediately implies that the growth rate of L_n is not elementary [3], meaning that L_n cannot be bounded by any function of the form $2^{2^{\cdot^{\cdot^{2^n}}}}$. In this subsection we show that L_n can be approximated by $2^{L_{n-1}} L_{n-1}$, that is $L_n = \Theta(2^{L_{n-1}} L_{n-1})$.

Corollary 3.3. *L_n and L_{n-1} satisfy the following inequalities for all $n \geq 2$.*

$$2^{L_{n-1}} L_{n-1} < L_n < 2^{L_{n-1}} \left(L_{n-1} + 4\log^2(L_{n-1}) \right).$$

Proof. It is easily checked that the inequalities hold for $n = 2$. Suppose $n \geq 3$. By using the inequality $L_n < 2^{S_{n-1}+1}$, which follows from (11), and the equality $S_{n-1} - S_{n-2} = L_{n-1}$ we derive the following.

$$\begin{aligned} L_n &\overset{(11)}{=} 2^{S_{n-1}+1} - 2^{S_{n-2}+1} - \sum_{i=1}^{S_{n-2}+1} 2^i L_{n-1}^i \;>\; 2^{S_{n-1}+1} - 2^{S_{n-2}+1} - L_{n-1} 2^{S_{n-2}+1} \\ &= 2^{S_{n-2}+1}(2^{L_{n-1}} - L_{n-1} - 1) \;\geq\; (L_{n-1}+1)(2^{L_{n-1}} - L_{n-1} - 1) \\ &\geq L_{n-1} 2^{L_{n-1}}. \end{aligned}$$

To establish the other inequality consider the quotient of L_n over L_{n-1}.

$$\begin{aligned} \frac{L_n}{L_{n-1}} &= \frac{2^{S_{n-1}+1} - 2^{S_{n-2}+1} - \sum_{i=1}^{S_{n-2}+1} 2^i L_{n-1}^i}{2^{S_{n-2}+1} - 2^{S_{n-3}+1} - \sum_{i=1}^{S_{n-3}+1} 2^i L_{n-2}^i} \\ &< \frac{2^{S_{n-1}+1}}{2^{S_{n-2}+1} - 2^{S_{n-3}+1} - L_{n-2} 2^{S_{n-3}+1}} \\ &= 2^{L_{n-1}} \left(1 + \frac{L_{n-2}+1}{2^{L_{n-2}} - L_{n-2} - 1} \right) \\ &\leq 2^{L_{n-1}} \left(1 + \frac{2L_{n-2}+2}{2^{L_{n-2}}} \right). \end{aligned}$$

Now $L_{n-1} \leq 2^{S_{n-2}+1} = 2^{S_{n-3}+1} \cdot 2^{L_{n-2}} \leq 2L_{n-2} 2^{L_{n-2}}$ follows from $2L_{n-2} > 2^{S_{n-3}+1}$. Thus

$$\frac{L_n}{2^{L_{n-1}} L_{n-1}} \;\leq\; 1 + \frac{2L_{n-2}+2}{\frac{L_{n-1}}{2L_{n-2}}} \;=\; 1 + \frac{4L_{n-2}(L_{n-2}+1)}{L_{n-1}} \;\leq\; 1 + \frac{4\log^2(L_{n-1})}{L_{n-1}}$$

using the inequalities $L_{n-1} \geq L_{n-2} 2^{L_{n-2}} \geq 2^{L_{n-2}+1}$. □

4. Regular C-graph

From the viewpoint of implementation, fixed-degree branching computational objects are much more manageable than the finite branching computational objects. We consider a subclass of C-graphs where the out-degrees of nodes are bounded by a constant. Recall that an internal node is a node that has at least one out-going edge to another node.

Definition 5. A C-graph is *k-regular* if the out-degree of every internal node is at most k.

The simplest regular C-graphs are 2-regular. Let $\mathcal{K}_n$ be the set of all 2-regular C-graphs in the n-th level and set $K_n = |\mathcal{K}_n|$. The set of all the 2-regular C-graphs that stay at or below the n-th level is $\mathcal{T}_n = \bigcup_{i=0}^{n} \mathcal{K}_i$. Let $T_n = |\mathcal{T}_n|$. It is not difficult to convince oneself that the derivation for (7) can be modified to establish the following equality that is valid for $n \geq 2$.

$$K_n = \binom{T_{n-1}+1}{2} - \binom{T_{n-2}+1}{2} - 2K_{n-1}.$$

The right hand side of the equality can be rewritten as follows:

$$\begin{aligned}\binom{T_{n-1}+1}{2}-\binom{T_{n-2}+1}{2}-2K_{n-1} &= \frac{1}{2}\left((T_{n-1}+1)T_{n-1}\right)-\frac{1}{2}\left((T_{n-2}+1)T_{n-2}\right)-2K_{n-1}\\ &= \frac{1}{2}(T_{n-1}+T_{n-2}+1)(T_{n-1}-T_{n-2})-2K_{n-1}\\ &= \frac{1}{2}(T_{n-1}+T_{n-2}+1)K_{n-1}-2K_{n-1}.\end{aligned}$$

It is now clear that $\frac{2K_n}{K_{n-1}}+3=T_{n-1}+T_{n-2}$ and also $\frac{2K_{n-1}}{K_{n-2}}+3=T_{n-2}+T_{n-3}$. So $\frac{2K_n}{K_{n-1}}=K_{n-1}+K_{n-2}+\frac{2K_{n-1}}{K_{n-2}}$. Hence

Corollary 4.1. *The following recursive equality holds for $n\geq 3$.*

$$K_n=\frac{K_{n-1}^2}{2}+\frac{K_{n-1}K_{n-2}}{2}+\frac{K_{n-1}^2}{K_{n-2}}.$$

Notice that $K_0=K_1=2$, $K_2=3$ and $K_3=12$. Thus $\frac{K_{n-1}^2}{K_{n-2}}\leq\frac{K_{n-1}^2}{3}$ for $n\geq 3$. Also notice that $K_3>3K_2$ and $K_n>\frac{K_{n-1}^2}{2}>\frac{12K_{n-1}}{2}>3K_{n-1}$ for $n\geq 4$. Therefore $3K_{n-1}K_{n-2}\leq K_{n-1}^2$ for all $n\geq 4$. It follows that $\frac{K_{n-1}K_{n-2}}{2}+\frac{K_{n-1}^2}{K_{n-2}}\leq\frac{K_{n-1}^2}{2}$ for all $n\geq 4$. Hence the following.

Corollary 4.2. $\frac{K_{n-1}^2}{2}\leq K_n\leq K_{n-1}^2$ *for all $n\geq 4$.*

We conclude that K_n is bounded by a double exponential function of n.

Now consider the general k-regular C-graphs. Let $\mathcal{K}_n^k$ be the set of all k-regular C-graphs in the n-th level and set $K_n^k=|\mathcal{K}_n^k|$. Let $\mathcal{K}_n^{k(i)}$ be the set of all k-regular C-graphs with the out-degree of the root being i and set $K_n^{k(i)}=|\mathcal{K}_n^{k(i)}|$. Let $\mathcal{T}_n^k=\bigcup_{i=1}^n\mathcal{K}_i^k$ and $T_n^k=|\mathcal{T}_n^k|$. Similar to the 2-regular case, the derivation of (7) can be repeated to show that

$$K_n^k=\sum_{j=1}^k\binom{T_{n-1}^k+1}{j}-\sum_{j=1}^k\binom{T_{n-2}^k+1}{j}-\sum_{i=1}^k 2^iK_n^{k(i)} \tag{14}$$

for all $n\geq 2$. Notice that

$$\sum_{i=1}^k 2^iK_n^{k(i)}\leq 2^k\sum_{i=1}^k K_n^{k(i)}=2^kK_n^k. \tag{15}$$

Using (14) and (15) we have

$$K_n^k\geq\sum_{j=1}^k\binom{T_{n-1}^k+1}{j}-\sum_{j=1}^k\binom{T_{n-2}^k+1}{j}-2^kK_n^k. \tag{16}$$

The following inequality follows immediately from (14) and (16).

$$\sum_{j=1}^k\binom{T_{n-1}^k+1}{j}-\sum_{j=1}^k\binom{T_{n-2}^k+1}{j}\leq(2^k+1)K_n^k\leq(2^k+1)\left(\sum_{j=1}^k\binom{T_{n-1}^k+1}{j}-\sum_{j=1}^k\binom{T_{n-2}^k+1}{j}\right). \tag{17}$$

By summing up the inequalities in (17) and bearing in mind that $T_1^k=5$, one obtains

$$\sum_{j=1}^k\binom{T_{n-1}^k+1}{j}-64\leq(2^k+1)T_n^k\leq(2^k+1)\sum_{j=1}^k\binom{T_{n-1}^k+1}{j}. \tag{18}$$

It follows from (18) and the following well-known bound

$$\left(\frac{n}{k}\right)^k\leq\binom{n}{k}\leq\left(\frac{en}{k}\right)^k$$

that

$$\frac{1}{2^k+1}\sum_{j=1}^k\left(\frac{T_{n-1}^k+1}{j}\right)^j-\frac{64}{2^k+1}\leq T_n^k\leq e^k\sum_{j=1}^k\left(\frac{T_{n-1}^k+1}{j}\right)^j. \tag{19}$$

The next theorem is an immediate consequence of (19).

Theorem 4.3. *For k-regular C-graphs and all $n \geq 2$, $c_1 \left(T^k_{n-1}\right)^k \leq T^k_n \leq c_2 \left(T^k_{n-1}\right)^k$ for some positive constants c_1, c_2 that depend on k.*

For fixed k the function T^k_n, and K^k_n as well is bounded by a double exponential function of n.

5. Remark on counting by graph size

Theorem 3.1 provides a nice and useful recursive equality for calculating the number of C-graphs counted level by level in terms of graph height. An obvious question is if there is a closed formula for S_n and L_n as well. Theory of enumerative combinatorics [26] might offer help in this regard. Since the focus of this paper is computational, we leave the issue for future investigation. In enumerative combinatorics, the usual approach is to count C-graphs according to the number of nodes. How many C-graphs of n nodes are there? The following theorem reveals the relationship between the height and the size of C-graph. Recall that $\log^*(n)$ is the minimal k such that $n \leq 2^{\cdot^{\cdot^{\cdot^{2}}}\}k}$.

Theorem 5.1. *The following statements are valid.*

1. *For $n \geq 3$, the height $\mathfrak{h}(\mathcal{G}_n)$ of an n-node C-graph $\mathcal{G}_n$ renders true the following.*
$$\log^*(n) - 1 \leq \mathfrak{h}(\mathcal{G}_n) \leq n - 2. \tag{20}$$
2. *The size $\mathfrak{s}(\mathcal{G}_h)$ of a non-trivial C-graph $\mathcal{G}_h$ of height h renders true the following.*
$$h + 2 \leq \mathfrak{s}(\mathcal{G}_h) \leq 1 + S_{h-1}. \tag{21}$$

Proof. It is not difficult to see that the C-graphs in Fig. 2 can be generalized to a C-graph of n nodes that is of height $n-2$. So the upper bound of (20) and the lower bound of (21) are tight. For the other inequality in (21) observe that an h-height C-graph may have all the nodes in the h'-th level for $h' < h$. Thus $\mathfrak{s}(\mathcal{G}_h) \leq 1 + S_{h-1}$. We now establish the lower bound in (20). It is easy to see that the inequality holds when $n = 3$, and the lower bound is tight. For $n \geq 4$, the height of an n-node C-graph is at least h if $n \geq 2 + S_{h-2}$ since a C-graph of height $h-1$ has at most $1 + S_{h-2}$ nodes. Next notice that we have the following bounds for L_h.

$$2^{S_{h-1}} < L_h < 2 \cdot 2^{S_{h-1}}. \tag{22}$$

The second inequality in (22) follows immediately from (11). For the first inequality, observe that

$$2^{S_{h-2}+1} + \sum_{j=1}^{S_{h-2}+1} 2^j L^j_{h-1} \leq 2^{S_{h-2}+1} + 2^{S_{h-2}+1} \sum_{j=1}^{S_{h-2}+1} L^j_{h-1} \leq 2^{S_{h-2}}(2 + 2L_{h-1}) \leq 2^{S_{h-2}} \cdot 2^{L_{h-1}} = 2^{S_{h-1}}.$$

Using (11) again we conclude that $L_h > 2^{S_{h-1}+1} - 2^{S_{h-1}} = 2^{S_{h-1}}$. It suffices to prove that $2^{n-2} \geq L_{h-1}$ when $h = \log^*(n) - 1$. This is because $2^{n-2} \geq L_{h-1} \geq 2^{S_{h-2}}$ by (22), which implies $n - 2 \geq S_{h-2}$. To establish the inequality $2^{n-2} \geq L_{h-1}$, we prove that the following is valid for all $h \geq 2$.

$$2^{\cdot^{\cdot^{\cdot^{2}}}\}h} \leq L_h \leq \frac{1}{4} \cdot 2^{\cdot^{\cdot^{\cdot^{2}}}\}h+2}. \tag{23}$$

The first inequality is due to Corollary 3.3. One has $L_h \geq L_{h-1} 2^{L_{h-1}} \geq 2^{L_{h-1}}$. We are done by natural induction, bearing in mind that $L_1 = 3$. The second inequality is derived from Corollary 3.3 and induction:

$$\begin{aligned} L_h &\leq 2^{L_{h-1}}(L_{h-1} + 4\log^2(L_{h-1})) \\ &\leq 2^{2L_{h-1}} \\ &\leq 2^{\frac{1}{2} \cdot 2^{\cdot^{\cdot^{\cdot^{2}}}\}h+1}} \\ &\leq \frac{1}{4} \cdot 2^{\cdot^{\cdot^{\cdot^{2}}}\}h+2}, \end{aligned}$$

where the last step is correct because $2^{\frac{1}{2}x} \leq \frac{1}{4}2^x$ for $x \geq 6$ and $L_2 = 40 > 6$. The base of the induction is $L_2 = 40 < \frac{1}{4}2^{2^{2^2}}$. It follows from (23) and definition that $2^n \geq 2^{\cdot^{\cdot^{\cdot^{2}}}\}h+1}$ with $h = \log^*(n) - 1$. We are done. □

Theorem 5.1 reveals that the difference between counting by size and counting by height is dramatic. Our initial investigation seems to suggest that the former is a lot trickier than the latter.

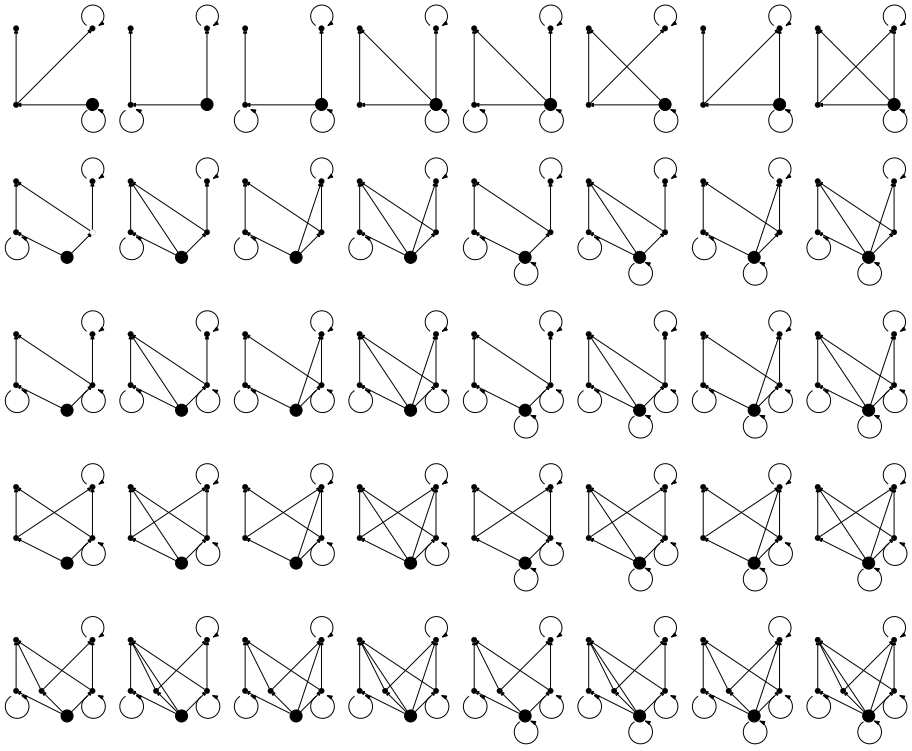

Fig. 5. C-Graphs of Height 2.

6. Comment

By casting computations in the framework of process theory, we are able to study the branching structure of nondeterministic computations in an abstract form. Our definition of the branching time complexity of a C-graph enables us to count the number of C-graphs. In a C-graph the root is the point an input is picked up, and the trivial node without loop is the point an output is released. In all other states the capability to engage in any interaction is absent at the point. This way of seeing the C-graphs helps explain the equality relation for them.

We have already observed that $L_0 = 2$ and $L_1 = 3$. Furthermore we obtain $L_2 = 40$ by Theorem 3.1. All the forty C-graphs are presented diagrammatically in Fig. 5. The number L_3 is staggering, it is $2^{46} - 676$. If 70 C-graphs of height three are drawn on one page, it takes 10^{12} pages to draw all of them. That says something about the quantitative aspect of nondeterminism.

Declaration of competing interest

There is no conflict of interest concerning the submission of this paper.

Acknowledgements

We thank NSFC (62072299, 61772336) for the financial support. We thank BASICS members for their interest. We also thank the anonymous referee for the comment and the insight, and for the observation that the non-elementary growth of S_n has a much simpler argument than we originally provided.

References

[1] L. Aceto, M. Hennessy, Termination, deadlock, and divergence, J. ACM 39 (1992) 147–187.
[2] A. Church, An unsolvable problem of elementary number theory, Am. J. Math. 58 (2) (1936) 345–363.
[3] N. Cutland, Computability: An Introduction to Recursive Function Theory, Cambridge University Press, 1980.
[4] P. van Emde Boas, Machine models and simulations, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science: Algorithm and Complexity, vol. A, Elsevier, 1990, pp. 65–116.
[5] Y. Fu, Nondeterministic structure of computation, Math. Struct. Comput. Sci. 25 (2015) 1295–1338.
[6] Y. Fu, Theory of interaction, Theor. Comput. Sci. 611 (2016) 1–49.
[7] Y. Fu, The universal process, Log. Methods Comput. Sci. 13 (2017) 1–23.
[8] Y. Fu, H. Lu, On the expressiveness of interaction, Theor. Comput. Sci. 411 (11–13) (2010) 1387–1451.

[9] R. van Glabbeek, W. Weijland, Branching time and abstraction in bisimulation semantics, in: Information Processing'89, North-Holland, 1989, pp. 613–618.
[10] R. van Glabbeek, B. Luttik, L. Spaninks, Rooted divergence-preserving branching bisimilarity is a congruence, arXiv:1801.01180, 2018.
[11] R. van Glabbeek, B. Luttik, N. Trčka, Branching bisimilarity with explicit divergence, Fundam. Inform. 93 (2009) 371–392.
[12] K. Gödel, Über Formal Unentscheidbare Sätze der Principia Mathematica und Verwandter Systeme, Monatshefte Math. Verwandter Syst. I 38 (1931) 173–198.
[13] D. Gorla, Comparing communication primitives via their relative expressive power, Inf. Comput. 206 (2008) 931–952.
[14] M. Hennessy, An Algebraic Theory of Processes, MIT Press, Cambridge, MA, 1988.
[15] C. Hoare, Communicating Sequential Processes, Prentice Hall, 1985.
[16] S.C. Kleene, Origins of recursive function theory, Ann. Hist. Comput. 3 (1) (1981) 52–67.
[17] M. Lohrey, P. D'Argenio, H. Hermanns, Axiomatising divergence, Inf. Comput. 203 (2005) 115–144.
[18] R. Milner, Communication and Concurrency, Prentice Hall, 1989.
[19] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes (part I), Inf. Comput. 100 (1992) 1–40;
R. Milner, J. Parrow, D. Walker, A calculus of mobile processes (part II), Inf. Comput. 100 (1992) 41–77.
[20] R. Milner, D. Sangiorgi, Barbed bisimulation, in: Proc. ICALP'92, in: Lecture Notes in Comput. Sci., vol. 623, 1992, pp. 685–695.
[21] U. Nestmann, Welcome to the jungle: asubjective guide to mobile process calculi, in: Proc. CONCUR'06, in: Lecture Notes in Comput. Sci., vol. 4137, 2006, pp. 52–63.
[22] C. Palamidessi, Comparing the expressive power of the synchronous and the asynchronous π-calculus, Math. Struct. Comput. Sci. 13 (2003) 685–719.
[23] D. Park, Concurrency and automata on infinite sequences, in: Theoretical Computer Science, in: Lecture Notes in Comput. Sci., vol. 104, Springer, 1981, pp. 167–183.
[24] L. Priese, On the concept of simulation in asynchronous, concurrent systems, Prog. Cybern. Syst. Res. 7 (1978) 85–92.
[25] D. Sangiorgi, D. Walker, The π Calculus: A Theory of Mobile Processes, Cambridge University Press, 2001.
[26] R.P. Stanley, Enumerative Combinatorics, vol. 2, Cambridge Stud. Adv. Math., 1999.
[27] A.M. Turing, On computable numbers, with an application to the Entscheidungsproblem, J. Math. 58 (345–363) (1936) 5.
[28] D. Walker, Bisimulation and divergence, Inf. Comput. 85 (1990) 202–241.