A Characterization of Regular Expressions under Bisimulation

J. C. M. BAETEN

Technische Universiteit Eindhoven, Eindhoven, The Netherlands

F. CORRADINI

Università di Camerino, Camerino, Italy

AND

C. A. GRABMAYER

Vrije Universiteit, Amsterdam, The Netherlands

Abstract. We solve an open question of Milner [1984]. We define a set of so-called well-behaved finite automata that, modulo bisimulation equivalence, corresponds exactly to the set of regular expressions, and we show how to determine whether a given finite automaton is in this set. As an application, we consider the star height problem.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata; relations between models*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*Interactive and reactive computation*

General Terms: Theory

Additional Key Words and Phrases: Bisimulation equivalence, regular expressions

ACM Reference Format:

Baeten, J. C. M, Corradini, F., and Grabmayer, C. A. 2007. A characterization of regular expressions under bisimulation. J. ACM 54, 2, Article 6 (April 2007), 28 pages. DOI = 10.1145/1219092.1219094 http://doi.acm.org/10.1145/1219092.1219094

1. *Introduction*

Automata and formal language theory have a place in every undergraduate computer science curriculum, as this provides students with a simple model of computation,

C. A. Grabmayer was supported by the Netherlands Organisation for Scientific Research NWO in project 612.000.313, GeoProc.

Contact Author's address: J. C. M. Baeten, Division of Computer Science, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, e-mail: josb@win.tue.nl.

and an understanding of computability. This simple model of computation does not include the notion of interaction, which is more and more important at a time when computers are always connected, and not many batch processes remain. Adding interaction to automata theory leads to concurrency theory:

$$\text{automata} + \text{interaction} = \text{concurrency}.$$

The basic ideas of concurrency theory should also be taught to computer science undergraduates, alongside automata theory. Then, it helps to consider similarities and differences between the two. Concurrency theory would benefit from an approach more along the lines of automata theory. We believe that this article, besides solving a long-standing open problem, will rekindle interest in a deeper study of the relationships between automata theory and concurrency theory. For one thing, concurrency theory knows a bewildering variety of different notations, and different composition operators. By using analogies with automata theory, some standardization could occur. For instance, if we all agree on a basic theory of regular processes, then basic concurrency theory would include the notions of alternative and sequential composition, and we could all use notations 0 and 1 for the basic constants.

In the past, frequently a stimulating exchange of ideas has taken place between automata and formal language theory, on the one hand, and concurrency theory, on the other hand. As examples, we mention the definition of a process interpretation for context-free languages that facilitated the discovery in Baeten et al. [1993] of a decidable process theory variant of the undecidable equivalence problem for context-free languages, and the use of a process calculus in Stirling [2001] for simplifying the proof of the celebrated result in Sénizergues [2001] that language equivalence is decidable for deterministic pushdown automata. However, in spite of a wealth of relationships that have been established, we believe that the correspondences between automata theory and concurrency theory have not yet been fully explored, and that the results have not been presented in an appealing, unified form.

A case in point is the question of how natural classes of processes can be characterized in terms of the interaction of paradigmatic processes with finite processes or with each other. We consider an example. In automata theory, there is the well-known correspondence between context-free grammars and pushdown automata under language equivalence. But in concurrency theory, only one half of this correspondence is recovered as the following inclusion: process interpretations of context-free grammars, which are recursively defined processes in Basic Process Algebra (recursively definable BPA-processes), only form a proper subclass of the class of processes that are described by transition graphs of pushdown automata ("pushdown processes"). However, a pushdown automaton can itself be represented as a process in which a finite automaton interacts with a stack. Using this fact, it is possible to show the following characterisation of pushdown processes: a process is a pushdown process if and only if it is equivalent to a finite-state process interacting with Stack, the paradigmatic stack process, which is a recursively definable BPA-process. In other words, the recursively definable BPA-process Stack is complete for the class of pushdown processes with respect to interaction with finite-state processes. This raises a host of new questions: Can the paradigmatic processes Bag (which models the nonterminating process of all possible addition and removal actions of copies of a finite set of objects to, and respectively from, a hidden storage) and Queue (which models a first-in first-out queue) also be used to

characterise interesting classes of processes? And do there exist interesting analoga for such results in automata theory?

A Turing machine can be viewed as a machine in which a finite control mechanism communicates with two copies of Stack simultaneously. This was used in Baeten et al. [1987] to show that every computable process is definable in the full concurrency theory ACP; similar results hold also for the well-known process theories CSP and CCS. However, the rôle of the Church–Turing thesis in process theory has not yet been investigated thoroughly. In particular, a general theory about which forms of communication are necessary to yield Turing-complete process models is still lacking. Therefore, a formulation of a version of the Church–Turing thesis that is specific to concurrency theory seems a worthwhile undertaking.

In formal language theory, there is a well-known correspondence between regular expressions and finite automata: for every regular expression a finite deterministic automaton (DFA) can be constructed that accepts the language denoted by the expression, and for every finite nondeterministic automaton (NFA) a regular expression can be produced that denotes the language accepted by the automaton. These transformations define correspondences between regular expressions and DFAs, and between regular expressions and NFAs, respectively.

But it is also well known that this correspondence of regular expressions under the language interpretation with DFAs and NFAs breaks down for Milner's process interpretation under bisimilarity, an equivalence that is finer than language equivalence. For most purposes in concurrency theory, language equivalence is not suitable, because only completed execution paths are compared: much information on intermediate states is lost, and hence interaction that may depend on transition options in intermediate states cannot be described adequately. Over the years, a number of other equivalences have been proposed that do not suffer from this drawback (see van Glabbeek [2001] for an overview). The equivalence that is most widely used and that keeps all information on transition options in intermediate states is bisimulation equivalence, also called bisimilarity. It has an intuitively appealing definition and can be characterized in many different ways.

In this article, we investigate regular expressions under bisimulation equivalence. By regular expressions, we mean expressions built with the operators $+$, $\cdot$, *. Sometimes, the operators complementation and intersection are included in the set of regular operators, but we do not do that here, referring to the situation where these extra operators are included as "generalized regular expressions".

Milner [1984] introduced a process interpretation of regular expressions as finite-state processes and showed that not every nondeterministic finite automaton (or finite-state process) is bisimulation equivalent to a regular expression, or equivalently, to a closed term in the process algebra with atomic actions, successful termination and deadlock, choice, sequential composition and iteration. Next to questions about an axiomatization of bisimilarity with respect to the process interpretation and about whether star height defines a hierarchy for regular expressions under bisimulation even over singleton alphabets, Milner [1984] posed the question:

> "What structural property of finite charts is necessary and sufficient for star behavior?"

This question can be reformulated in terms of finite-state processes (instead of relying on the comparable concept "chart" defined by Milner) and by avoiding the notions "behavior" (a behavior is a bisimulation equivalence class of processes)

and "star behavior" (a bisimulation equivalence class of the process interpretation of a regular expression) as follows:

> What structural property characterizes those finite-state processes that are expressible as regular expressions under Milner's process interpretation?

By saying that a process is "expressible as a regular expression under the process interpretation" we mean that this process is bisimilar to the process interpretation of a regular expression. Here, we solve this open question: we show that there exists a decision algorithm for determining whether or not a finite-state process is expressible by a regular expression under the process interpretation. Our solution is based on, and extends, Baeten and Corradini [2005], where two of the present authors have defined a set of recursive specifications, called "well behaved", and shown that these specifications correspond exactly to processes that are expressible as regular expressions. Here we complement that result by proving that it is decidable whether or not a finite-state process is the solution of a well-behaved specification. In this way, we establish that expressibility by a regular expression under the process interpretation is decidable.

As an application of the methods used in our decidability proof that employ reasoning about well-behaved specifications, we show that the star-height problem under the process interpretation is solvable: we give an algorithmic solution to the problem of finding, for a given regular expression e, the least natural number n such that there exists a regular expression f of star height n with the property that e and f have bisimilar process interpretations. Finally, as another instance of reasoning with well-behaved specifications, we give an alternative proof for a result of Hirshfeld and Moller [2000] that solved the star-height question posed by Milner: for every natural number n there exists a regular expression over a singleton alphabet that is not bisimilar to any regular expression of star height less than n.

2. *Process Algebra*

We start out from the equational theory $\mathrm{BPA}^*_{0,1}$. Closed terms in this theory correspond exactly to the regular expressions of formal language theory. We use notations from regular expressions mainly, but want to emphasise the fact that we consider bisimulation equivalence as our notion of equivalence, and not language equivalence. $\mathrm{BPA}^*_{0,1}$ extends the basic process algebra BPA (see Bergstra and Klop [1984]) with constants 0 and 1 and iteration operator $*$. We assume we have given a set of actions A. This set, usually (but not necessarily) finite, is considered a parameter of the theory. The signature elements are:

—Binary operator $+$ denotes *alternative composition* or choice. Process $x + y$ executes either x or y, but not both. The choice is resolved upon execution of the first action. The notation $+$ is also used for regular expressions.

—Binary operator $\cdot$ denotes *sequential composition*. We choose to have sequential composition as a basic operator, different from CCS (see Milner [1989]). As a result, we have a difference between successful termination (1) and deadlock (0). As is done for regular expressions, this operator is sometimes not written.

TABLE I. AXIOMS OF $\mathrm{BPA}^*_{0,1}$

$x + y = y + x$	A1
$(x + y) + z = x + (y + z)$	A2
$x + x = x$	A3
$(x + y) \cdot z = x \cdot z + y \cdot z$	A4
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5
$x + 0 = x$	A6
$0 \cdot x = 0$	A7
$1 \cdot x = x$	A8
$x \cdot 1 = x$	A9
$x^* = 1 + x \cdot x^*$	KS1
$(x + 1)^* = x^*$	KS2
$x^* \cdot (y \cdot (x + y)^* + 1) = (x + y)^*$	KS3

—Constant 0 denotes *inaction* (or deadlock), and is the neutral element of alternative composition. This constant is denoted δ in ACP-style process algebra [Baeten and Weijland 1990], denoted 0 or *nil* in CCS-style process algebra [Milner 1989], and denoted *stop* in CSP-style process algebra [Hoare 1985]. Process 0 cannot execute any action, and cannot terminate. In language theory, notations 0 and $\emptyset$ are used.

—Constant 1 denotes the *empty process* or *skip*. It is the neutral element of sequential composition. This constant is denoted ϵ in ACP-style process algebra [Baeten and Weijland 1990] and *skip* in CSP-style process algebra [Hoare 1985]. Process 1 cannot execute any action, but terminates successfully. The notation 1 is also used in language theory.

—We have a constant a for each $a \in A$, a so-called *atomic action*. Process a executes action a and then terminates successfully. This coincides with the notation in language theory. The set of actions A is considered a parameter of the theory.

—There is a unary operator $*$ called *iteration* or *Kleene star*. Process x^* can execute x any number of times, but can also terminate successfully. This coincides with the notation in language theory. In Bergstra et al. [1994], a *binary* version of this operator is used. We can use the unary version, common in language theory, as we have a constant 1.

The equational theory $\mathrm{BPA}^*_{0,1}$ is given by axioms A1-9 and KS1-3 in Table I. Axioms A1-9 are standard. Compared to language theory, we do not have the law $x \cdot (y + z) = x \cdot y + x \cdot z$. This axiom can be called the "wrong" distributivity, as the terms differ in the moment of choice. We also do not have the law $x \cdot 0 = 0$; thus, 0 is not a "real" zero: in $x \cdot 0$, actions from x can be executed but no termination can take place, whereas in 0, no action at all can be executed.

KS1 defines iteration in terms of a recursive equation. Taking $x = 0$ yields $0^* = 1$. KS2 expresses that immediate termination can be omitted in iteration behavior. In language theory, we say that we can assume that the iterated term does not have the empty word property. We will also use this terminology, so we say that term t has the empty word property iff it is derivable that $t = t + 1$. Taking $x = 0$ in axiom KS2 and using $0^* = 1$ yields $1^* = 1$. KS3 is an axiom that stems from Troeger [1993].

The regular expressions are the *closed* terms over this theory, that is, the terms without variables. Many results in process algebra, like the following normal form lemma, only hold on the set of closed terms.

Definition 2.1. We define a set of normal forms inductively:

(1) the constants 0, 1 are normal forms;
(2) if t, s are normal forms, and a is an atomic action, then also $a \cdot t$ and $t + s$ are normal forms;
(3) if t, s are normal forms, and t does not have the empty word property (i.e., $t = t + 1$ is not derivable), then $t^* \cdot s$ is a normal form.

PROPOSITION 2.2. *Let t be a closed $BPA^*_{0,1}$-term. There is an effective algorithm producing a normal form s such that $BPA^*_{0,1} \vdash t = s$.*

PROOF. First of all, every closed $\mathrm{BPA}^*_{0,1}$-term t can be transformed into a term $\tilde{t}$ such that $\mathrm{BPA}^*_{0,1} \vdash t = \tilde{t}$ holds and for each subterm u^* of $\tilde{t}$, u does not have the empty word property (Proposition 6.2 in Milner [1984]). This is a consequence of the possibility to effectively replace in a given closed $\mathrm{BPA}^*_{0,1}$-term t, in a top-down manner, starred subterms u^* where u has the empty word property by starred subterms $\tilde{u}^*$ such that $\tilde{u}$ does not have the empty word property, resulting in a closed $\mathrm{BPA}^*_{0,1}$-term $\tilde{t}$ as required. Such replacements are possible due to the presence of the axiom KS2 in $\mathrm{BPA}^*_{0,1}$ as well as due to the following fact, which can be established in a straightforward manner by structural induction on e : for every closed $\mathrm{BPA}^*_{0,1}$-term e that has the empty word property, a closed $\mathrm{BPA}^*_{0,1}$-term $\tilde{e}$ can be built such that $\mathrm{BPA}^*_{0,1} \vdash e = 1 + \tilde{e}$ holds, $\tilde{e}$ does not have the empty word property, and $\tilde{e}$ does not have greater nesting of * than e.

Next, turn the axioms A3-9 of $\mathrm{BPA}^*_{0,1}$ into rewrite rules, by orienting them from left to right. This gives a *term rewriting system* [TeReSe 2003]. As this is a confluent and terminating term rewrite system modulo A1-2, every closed term by rewriting reaches a unique normal form, that cannot be rewritten any further. Given a closed term t, its normal form may still contain summands of the form a (only an atomic action) or u^* (only an iteration). These have to be replaced by $a \cdot 1$ and $u^* \cdot 1$, respectively, thereby obtaining a normal form as defined above. Note that each rewriting preserves that iterated terms do not have the empty word property. This proof is like several examples in Baeten and Weijland [1990] or Baeten and Verhoef [1995]. □

As a consequence of this proposition, each closed term over $\mathrm{BPA}^*_{0,1}$ can be written as 0, 1 or in the form

$$a_1 \cdot t_1 + \cdots + a_n \cdot t_n + u_1^* \cdot v_1 + \cdots + u_m^* \cdot v_m + \{1\},$$

for certain $n, m \in \mathbb{N}$ with $n + m > 0$, certain $a_i \in A$ and normal forms t_i, u_j, v_j such that each u_j does not have the empty word property. The 1 summand may or may not occur. The additional requirement that terms u_j do not have the empty word property will ensure that the recursive specifications we define below are guarded.

A *model* of an equational theory is a mathematical structure such that, using a certain interpretation of the syntax, all equations of the theory are true in the structure. Elements of a model of an equational theory in concurrency theory are called *processes*. In language theory, the set of computations (complete runs) can be used as a model for the equational theory of regular expressions. This set of computations can be obtained from an automaton. Stated in another way, a model is obtained as language equivalence classes of automata.

TABLE II. TRANSITION RULES FOR $\mathrm{BPA}^*_{0,1}$ ($a \in A$)

$1\downarrow$	$x^*\downarrow$	$a \xrightarrow{a} 1$
$\dfrac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'}$	$\dfrac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$	
$\dfrac{x\downarrow}{x + y\downarrow}$	$\dfrac{y\downarrow}{x + y\downarrow}$	
$\dfrac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y}$	$\dfrac{x\downarrow,\ y \xrightarrow{a} y'}{x \cdot y \xrightarrow{a} y'}$	$\dfrac{x\downarrow,\ y\downarrow}{x \cdot y\downarrow}$
$\dfrac{x \xrightarrow{a} x'}{x^* \xrightarrow{a} x' \cdot x^*}$		

In concurrency theory, nondeterministic automata are usually called *transition systems*. Note that also infinite-state transition systems are considered. Language equivalence classes of transition systems also yield a model of $\mathrm{BPA}^*_{0,1}$, but this model is not preferred as too much is true: we have the wrong distributivity and the 0 process is a real zero, and this throws away too much information of the underlying transition systems: we want to keep information of intermediate states. Therefore, bisimulation equivalence is used instead.

So, we provide a model for $\mathrm{BPA}^*_{0,1}$ by first associating a transition system to each term, and then taking the quotient with respect to bisimulation equivalence. In concurrency theory, there is a standard way to associate a transition system to a closed term, called *Structural Operational Semantics*. This technique was initiated by Plotkin [2004]. Much more information can be found in Aceto et al. [2001].

Structural Operational Semantics (SOS) consists of providing rules for each operator and constant of the equational theory. The rules in Table II define the following relations on closed $\mathrm{BPA}^*_{0,1}$-terms: binary relations $. \xrightarrow{a} .$ (for $a \in A$) and a unary relation $\downarrow$. Intuitively, they have the following meaning:

—$x \xrightarrow{a} x'$ means that x can evolve into x' by executing atomic action a;
—$x\downarrow$ means that x has an option to terminate successfully (without executing an action)

Thus, the relations concern action execution and termination, respectively; we do not have need of a mixed relation $. \xrightarrow{a} \surd$ as in Baeten and Weijland [1990] or Baeten and Verhoef [1995].

The first three rules are axioms: they state that process 1 and every iteration has a termination option, and that process a has an a-labeled transition to 1. The next two lines state that process $x + y$ has all transitions and terminations of x and of y, so the possibilities are joined. The first step of $x \cdot y$ can be a first step of x, or a first step of y if x can terminate. Process $x \cdot y$ can only terminate if both x and y can do this. Finally, the last rule shows the first step of an iteration.

For a closed term, a transition system can be defined by using the transition rules: $t \xrightarrow{a} s$ or $t\downarrow$ holds iff this is provable from the rules in Table II. By structural induction on closed terms, it can be shown that the transitions of a given closed term can be determined algorithmically. Next, we define an equivalence relation on the resulting transition systems in the standard way.

Definition 2.3. Let R be a binary *symmetric* relation on closed terms. We say R is a *bisimulation* if the following holds:

—whenever $R(x, y)$ and $x \xrightarrow{a} x'$, then there is a term y' such that $y \xrightarrow{a} y'$ and $R(x', y')$

—whenever $R(x, y)$ and $x \downarrow$, then $y \downarrow$.

We say two closed terms t, s are *bisimulation equivalent* or *bisimilar*, notation $t \leftrightarrows s$ if there is a bisimulation R with $R(s, t)$.

PROPOSITION 2.4. *Bisimulation equivalence is a congruence relation on closed* $BPA^*_{0,1}$*-terms.*

PROOF. This is a standard result following from the format of the transition rules, see, for example, Baeten and Verhoef [1995]. □

THEOREM 2.5. *The theory* $BPA^*_{0,1}$ *is sound for the model of transition systems modulo bisimulation, that is, for all closed terms* t, s *we have*

$$BPA^*_{0,1} \vdash t = s \quad \Longrightarrow \quad t \leftrightarrows s.$$

PROOF. This is also a standard result. □

Note that the reverse implication in the theorem above, indicating completeness of the axiom system, does not hold. In fact, a finite complete equational axiomatization is not possible, as shown by Sewell [1997]. This impossibility is due to the combination of iteration and the 0 constant. In the absence of iteration, completeness is straightforward, see, for example, Baeten and Weijland [1990].

In the presence of iteration but without 0, more positive results can be found in Corradini [2000] and Corradini et al. [2002] where a complete axiomatization of regular expressions up to bisimulation equivalence is given when the language of regular expressions satisfies the so-called hereditary nonempty word property (essentially requiring that the nonempty word property be satisfied at any depth within a star context).

The previous theorem is a result about the term model, that is, the model generated by SOS rules for closed $\mathrm{BPA}^*_{0,1}$-terms only. In the sequel, we will have occasion to go beyond this model, in particular by adding *recursion*. Recursion is considered in the form of *recursive equations*. Recursive equations are a standard way to specify processes with possible infinite behavior, see, for example, Baeten and Weijland [1990] or Baeten and Verhoef [1995]. The first axiom of iteration is an example of a recursive equation, as the item to be defined (x^*), the left-hand side, occurs again on the right-hand side.

We proceed to define recursive equations in our setting. In language theory, a recursive specification would be called a *grammar*, but there, finiteness is always required.

Let V be a set of variables ranging over processes. A *recursive specification* $E = E(V)$ is a set of equations $E = \{X = t_X \mid X \in V\}$, where each t_X is a term over the signature in question (in our case, $\mathrm{BPA}^*_{0,1}$) and variables from V. A *solution* of a recursive specification $E(V)$ in our theory is a family of processes $\{p_X \mid X \in V\}$ in some model of the theory such that the equations of $E(V)$ hold, if for all $X \in V$, p_X is substituted for X. Mostly, we are interested in one particular variable $X \in V$, called the *initial* variable.

TABLE III. TRANSITION RULES FOR RECURSION ($a \in A$)

$\dfrac{\langle t_X \mid E\rangle \xrightarrow{a} y}{\langle X \mid E\rangle \xrightarrow{a} y}$	$\dfrac{\langle t_X \mid E\rangle \downarrow}{\langle X \mid E\rangle \downarrow}$

Different models extending the term model introduced above, will have solutions for different sets of recursive specifications. We proceed to define interesting sets of recursive specifications.

Let t be a term containing a variable X. We call an occurrence of X in t *guarded* if this occurrence of X is preceded by an atomic action (i.e., t has a subterm of the form $a \cdot s$, and this X occurs in s).

We call a recursive specification *guarded* if all occurrences of all its variables in the right-hand sides of all its equations are guarded or it can be rewritten to such a recursive specification using the axioms of the theory and the equations of the specification.

We can formulate the following principles:

—RDP (the *Recursive Definition Principle*): Each recursive specification has at least one solution;

—RSP (the *Recursive Specification Principle*): Each *guarded* recursive specification has at most one solution.

Different models of $\mathrm{BPA}^*_{\delta,1}$ will satisfy none, one or both of these principles. Let us look at particular models extending the term model of transition systems modulo bisimulation.

Consider a recursive specification E. For each variable X of E, we can add a new constant $\langle X \mid E\rangle$ to our syntax. Table III provides transition rules for these constants. The operational meaning of a constant $\langle X \mid E\rangle$ is defined as that of the process $\langle t_X \mid E\rangle$, which is t_X with, for all $Y \in V$, all occurrences of Y in t_X replaced by $\langle Y \mid E\rangle$. To be more explicit, if E is a *finite* recursive specification over variables $X_0, \ldots, X_{n-1}$, with equations $X_i = t_i(X_0, \ldots, X_{n-1})$ $(i < n)$, then $\langle t_i \mid E\rangle$ is defined to be $t_i(\langle X_0 \mid E\rangle, \ldots, \langle X_{n-1} \mid E\rangle)$.

Now if we add such constants $\langle X \mid E\rangle$ for all recursive specifications E to our syntax, and take the model of transition systems generated by the rules modulo bisimulation, we obtain a model $\mathbb{G}^\infty$ for $\mathrm{BPA}^*_{\delta,1}$ that satisfies RDP and RSP (see, e.g., Baeten and Weijland [1990]). If we add constants $\langle X \mid E\rangle$ only for guarded E, we obtain a model $\mathbb{G}$ satisfying RSP. The model $\mathbb{G}$ is really smaller than $\mathbb{G}^\infty$, since using unguarded recursion we can specify infinitely branching processes (infinitely branching means that *every* element of the bisimulation equivalence class is infinitely branching), whereas for guarded recursion we can always get a finitely branching solution (the bisimulation equivalence class contains at least one element that has only finite branching). Thus, RDP doesn't hold any more on the second model in full generality; still, all guarded recursive specifications have a solution. We call this RDP^-:

—RDP^- (the *Restricted Recursive Definition Principle*): Each guarded recursive specification has at least one solution;

The set of processes definable by a finite guarded recursive specification over $\mathrm{BPA}^*_{0,1}$ can be called the set of *context-free* processes. Of course, since the star operator is itself definable by a finite guarded recursive specification, this is the same as the set of processes definable by a finite guarded recursive specification over $\mathrm{BPA}_{0,1}$. Sometimes, the set of processes definable by a finite guarded recursive specification over BPA is called the set of context-free processes. It does make a difference whether we consider $\mathrm{BPA}_{0,1}$ or the standard theory BPA, since finite guarded recursion including the constant 1 allows the specification of a process with unbounded branching (see Bosscher [1997]). Context-free processes were investigated in Baeten et al. [1993] and Hirshfeld and Moller [1996].

Thus, we looked at the model obtained by adding *all* constants $\langle X|E\rangle$, the model obtained by adding constants $\langle X|E\rangle$ only for guarded E, and the model obtained by adding these constants only for finite guarded E. The fourth possibility is adding still fewer constants, adding only $\langle X|E\rangle$ for so-called *regular* recursive specifications. We call an equation *regular* if it is in one of the following two forms

(1) $X = (0+)a_1 \cdot X_1 + \cdots + a_n \cdot X_n$,
(2) $X = 1 + a_1 \cdot X_1 + \cdots + a_n \cdot X_n$,

for certain $n \in \mathbb{N}$, $a_i \in A$, $X_i \in V$. In this case, each variable corresponds directly to a state in the generated transition system. We usually present a regular equation as follows:

$$X = \sum_{1 \le i \le n} a_i \cdot X_i + \{1\},$$

where an empty sum stands for 0 and the 1 summand is optional.

The model $\mathbb{R}$ of $\mathrm{BPA}^*_{0,1}$ is obtained if we add constants $\langle X|E\rangle$ only for finite regular E. $\mathbb{R}$ is the model of *regular* processes, it is equivalent to the model of finite transition systems modulo bisimulation, see, for example, Bergstra and Klop [1984]. This result is the same result as the result in language theory, that right-linear grammars correspond to nondeterministic finite automata. Again we can establish that $\mathbb{R}$ is really smaller than $\mathbb{G}$, a process in the difference is the counter C defined by the following specification (p standing for plus, m for minus):

$$C = T \cdot C \qquad T = p \cdot S \qquad S = m + T \cdot S.$$

This process is in the difference, as every transition system in its bisimulation equivalence class must have infinitely many states (there is a different state for every counter value).

Finally, the term model $\mathbb{P}$ of $\mathrm{BPA}^*_{0,1}$, obtained by adding no constants $\langle X|E\rangle$, is even smaller than $\mathbb{R}$. In Bergstra et al. [1994], it is shown that there are regular processes that cannot be defined just by using iteration. In the model $\mathbb{P}$, the principle RSP boils down to the following conditional axiom:

$$x = y \cdot x + z \text{ guarded} \quad \Longrightarrow \quad x = y^* \cdot z \qquad \text{RSP*}.$$

The guardedness of this equation would be expressed in language theory as follows: y does not have the empty word property (y does not have 1 as a summand). Operationally, this is denoted $y \not\downarrow$, so we will use the following formulation of RSP*:

$$x = y \cdot x + z \ \ \& \ \ y \not\downarrow \quad \Longrightarrow \quad x = y^* \cdot z \qquad \text{RSP*}.$$

It is an open problem whether the addition of principle RSP* to the axiomatization $\mathrm{BPA}^*_{0,1}$ provides a complete axiomatization of the model $\mathbb{P}$.

3. *Well-Behaved Specifications*

We define a class of recursive specifications over $\mathrm{BPA}^*_{0,1}$ that we will call *well-behaved*. The idea is that the class of well-behaved specifications corresponds exactly to the class of closed $\mathrm{BPA}^*_{0,1}$-terms. In this section, after some preliminary definitions, we define well-behaved specifications and prove that each well-behaved specification has a closed $\mathrm{BPA}^*_{0,1}$-term as a solution, in the following section we prove the reverse direction: how to provide a well-behaved specification for each closed $\mathrm{BPA}^*_{0,1}$-term.

Consider sequences of natural numbers ranged over by σ, ρ (sometimes with a prime or index) ($\sigma, \rho \in \mathbb{N}^*$). Call a subset S of $\mathbb{N}^*$ *downwards closed* if the empty sequence $\epsilon \in S$, and, whenever $\sigma n \in S$, also $\sigma \in S$ and $\sigma k \in S$ for all $k < n$.

Definition 3.1. A recursive specification E over $\mathrm{BPA}^*_{0,1}$ is *in suitable form* if

(1) it is finite and guarded;
(2) the set of variables X_σ is indexed by a downwards closed subset of $\mathbb{N}^*$;
(3) each equation has the following form:
$$X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \cdots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho + c,$$
where $m \geq 0$, $e_{\sigma i}$, e_σ, e_ρ are closed $\mathrm{BPA}^*_{0,1}$-terms, $c \in \{0, 1\}$, and ρ is a proper prefix of σ. The last three terms may or may not be present (of course, if $m = 0$, at least one of them must be present). If there is a summand of the form $e_\rho \cdot X_\rho$ present, we call the variable X_ρ a *cycling* variable;
(4) when X_ρ is a cycling variable (it occurs in the right-hand side of an equation of a variable with longer index) then its equation is of the form
$$X_\rho = 1 \cdot X_{\rho 0} + 1 \cdot X_{\rho 1},$$
that is, $m = 2$, $e_{\rho 0} = e_{\rho 1} = 1$ and none of the optional summands is present.

A recursive specification is *in regular suitable form* if it is in suitable form and all the occurring closed terms e_σ are constants, that is, elements of $A \cup \{0, 1\}$.

The variables in a recursive specification in suitable form can be arranged as a finite tree, with X_ϵ as root, and such that X_σ is below X_ρ iff ρ is a prefix of σ.

In addition to the definition of suitable form, we need to define the notion of "cycling back". Intuitively, a variable X_σ cycles back to X_ρ if X_σ lies on a cycle (iteration behavior) that is initiated at X_ρ. Finally, we define the notion "well behaved", that will give us the specifications that we want. A cycling variable X_ρ in a well-behaved specification has an equation $X_\rho = 1 \cdot X_{\rho 0} + 1 \cdot X_{\rho 1}$, and $X_{\rho 0}$ is the cycling part, in which variables will cycle back to X_ρ, whereas $X_{\rho 1}$ is the exit part, where variables may terminate, or cycle back to a cycling variable higher in the tree, indexed by a prefix of ρ.

Definition 3.2. Let E be a recursive specification in suitable form over a set of variables $\{X_\sigma : \sigma \in S \subset \mathbb{N}^*\}$. As S is finite, we can define a notion with induction on the depth of the variable tree below X_σ (so, we define this first for the *maximal* sequences $\sigma \in S$).

Let ρ be a prefix of σ. We say X_σ *cycles back to* X_ρ if:

—in case X_σ is cycling (so its equation is of the form $X_\sigma = 1 \cdot X_{\sigma 0} + 1 \cdot X_{\sigma 1}$), then X_σ cycles back to X_ρ iff $X_{\sigma 0}$ cycles back to X_σ and $X_{\sigma 1}$ cycles back to X_ρ;
—in case X_σ is not cycling, we require that its equation is of the form

$$X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \cdots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho,$$

so there is no constant term present, the last two summands are optional, and all $X_{\sigma i}$ cycle back to X_ρ. In case $m = 0$, we must have that the last summand is present.

Next, we define when a variable X_σ is well behaved, again with induction on the depth of the variable tree below X_σ. We say X_σ is *well behaved* if:

—in case X_σ is cycling, we say X_σ is well behaved iff $X_{\sigma 0}$ cycles back to X_σ and $X_{\sigma 1}$ is well-behaved;
—in case X_σ is not cycling, we require that its equation is of the form

$$X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \cdots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + c,$$

so there is no cycling variable present, the next to last summand is optional, and all $X_{\sigma i}$ are well behaved.

Finally, we call a recursive specification E in suitable form *well behaved* iff its initial variable X_ϵ is well behaved.

THEOREM 3.3. *Every well-behaved recursive specification E has a closed term in $\mathrm{BPA}^*_{0,1}$ as a solution (up to bisimulation equivalence).*

In order to prove this theorem, we prove two lemmas. The theorem will follow from the two lemmas.

In these lemmas, we will use the following notation. Write $X_\sigma \searrow X_\rho$ if the equation of X_σ contains a summand $e_\rho \cdot X_\rho$ with $e_\rho \downarrow$. $\searrow\!\!\!\!\searrow$ denotes the transitive closure of the relation $\searrow$ on recursion variables. A useful characterization of guardedness is the following: a recursive specification in suitable form is guarded if and only if for each variable X_σ, we have that $X_\sigma \not\searrow\!\!\!\!\searrow X_\sigma$.

In the lemmas, we reason in the theory $\mathrm{BPA}^*_{0,1} + \mathrm{RSP}^*$. So, given a recursive specification E, when we derive an equation $s = t$, we mean $\mathrm{BPA}^*_{0,1} + \mathrm{RSP}^* + E \vdash s = t$. If variables from E occur in s, t, this means $s = t$ holds for any solution of E. As a result, if $\mathbb{M}$ is any model of the theory $\mathrm{BPA}^*_{0,1} + \mathrm{RDP}^- + \mathrm{RSP}^*$, and E has a solution in $\mathbb{M}$, then $s = t$ in true in $\mathbb{M}$.

LEMMA 3.4. *Let E be a recursive specification in suitable form, and suppose X_σ cycles back to X_ρ. Then there is a closed term e over $\mathrm{BPA}^*_{0,1}$ such that $\vdash X_\sigma = e \cdot X_\rho$. Moreover, if $X_\sigma \not\searrow\!\!\!\!\searrow X_\rho$, we can take e such that $e \not\downarrow$.*

PROOF. By induction on the depth of the variable tree below X_σ.

In the base case, there are no variables below X_σ, so the equation of X_σ must be either $X_\sigma = e_\rho \cdot X_\rho$ or $X_\sigma = e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho$. By guardedness, $e_\sigma \not\downarrow$. In the first case, we are done immediately, in the second case, it follows from RSP* that $X_\sigma = e^*_\sigma \cdot e_\rho \cdot X_\rho$. When $X_\sigma \not\searrow\!\!\!\!\searrow X_\rho$, we must have $e_\rho \not\downarrow$, which implies $e^*_\sigma \cdot e_\rho \not\downarrow$.

In the induction case, there are two subcases.

—if X_σ is cycling, we have $X_\sigma = 1 \cdot X_{\sigma 0} + 1 \cdot X_{\sigma 1}$ and $X_{\sigma 0}$ cycles back to X_σ and $X_{\sigma 1}$ cycles back to X_ρ. Since $X_\sigma \searrow X_{\sigma 0}$, we must have $X_{\sigma 0} \not\searrow\!\!\!\!\searrow X_\sigma$. We can

apply the induction hypothesis to $X_{\sigma 0}$ and $X_{\sigma 1}$, so there are closed terms f_0, f_1 such that $X_{\sigma 0} = f_0 \cdot X_\sigma$ and $X_{\sigma 1} = f_1 \cdot X_\rho$ and $f_0 \not\downarrow$. Putting this together, we obtain $X_\sigma = 1 \cdot X_{\sigma 0} + 1 \cdot X_{\sigma 1} = X_{\sigma 0} + X_{\sigma 1} = f_0 \cdot X_\sigma + f_1 \cdot X_\rho = f_0^* \cdot f_1 \cdot X_\rho$. When $X_\sigma \not\trianglerighteq X_\rho$, then $X_{\sigma 1} \not\trianglerighteq X_\rho$, and we can take $f_1 \not\downarrow$, which implies $f_0^* \cdot f_1 \not\downarrow$.

—if X_σ is not cycling, we have $X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \cdots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho$ and all $X_{\sigma i}$ cycle back to X_ρ. Again, $e_\sigma \not\downarrow$. By induction hypothesis there are closed terms f_i such that $X_{\sigma i} = f_i \cdot X_\rho$. But then $X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \cdots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho = e_{\sigma 0} \cdot f_0 \cdot X_\rho + \cdots + e_{\sigma(m-1)} \cdot f_{m-1} \cdot X_\rho + e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho = e_\sigma \cdot X_\sigma + (e_{\sigma 0} \cdot f_0 + \cdots + e_{\sigma(m-1)} \cdot f_{m-1} + e_\rho) \cdot X_\rho = e_\sigma^* \cdot (e_{\sigma 0} \cdot f_0 + \cdots + e_{\sigma(m-1)} \cdot f_{m-1} + e_\rho) \cdot X_\rho$. When $X_\sigma \not\trianglerighteq X_\rho$, then $e_\rho \not\downarrow$, and moreover for each $i < m$ we have either $e_{\sigma i} \not\downarrow$ or $X_{\sigma i} \not\trianglerighteq X_\rho$. In the latter case we can take $f_i \not\downarrow$, so, in all cases, we have $e_{\sigma i} \cdot f_i \not\downarrow$. Consequently $e_{\sigma 0} \cdot f_0 + \cdots + e_{\sigma(m-1)} \cdot f_{m-1} + e_\rho \not\downarrow$ and so $e_\sigma^* \cdot (e_{\sigma 0} \cdot f_0 + \cdots + e_{\sigma(m-1)} \cdot f_{m-1} + e_\rho) \not\downarrow$. □

LEMMA 3.5. *Let E be a recursive specification in suitable form, and suppose variable X_σ is well behaved. Then, there is a closed term e over $BPA^*_{0,1}$ such that $\vdash X_\sigma = e$.*

PROOF. By induction on the depth of the variable tree below X_σ.

In the base case, there are no variables below X_σ, so the equation of X_σ must be either $X_\sigma = c$ or $X_\sigma = e_\sigma \cdot X_\sigma + c$ or $X_\sigma = e_\sigma \cdot X_\sigma$. Note $e_\sigma \not\downarrow$. In the first case, we are done immediately, in the second case, it follows from RSP* that $X_\sigma = e_\sigma^* \cdot c$; and in the third case, $X_\sigma = e_\sigma^* \cdot 0$.

In the induction case, there are two subcases.

—if X_σ is cycling, then $X_\sigma = 1 \cdot X_{\sigma 0} + 1 \cdot X_{\sigma 1}$ and $X_{\sigma 0}$ cycles back to X_σ and $X_{\sigma 1}$ is well behaved. By Lemma 3.4, there is a closed term f such that $X_{\sigma 0} = f \cdot X_\sigma$. As $X_\sigma \searrow X_{\sigma 0}$, we must have $X_{\sigma 0} \not\trianglerighteq X_\sigma$, and we can take $f \not\downarrow$. By induction hypothesis, there is a closed term f' such that $X_{\sigma 1} = f'$. Thus, $X_\sigma = X_{\sigma 0} + X_{\sigma 1} = f \cdot X_\sigma + f' = f^* \cdot f'$.

—if X_σ is not cycling, we have $X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \cdots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + c$ and all $X_{\sigma i}$ are well behaved. By induction hypothesis this implies that there are closed terms f_i such that $X_{\sigma i} = f_i$. Thus, $X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \cdots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + c = e_{\sigma 0} \cdot f_0 + \cdots + e_{\sigma(m-1)} \cdot f_{m-1} + e_\sigma \cdot X_\sigma + c = e_\sigma^* \cdot (e_{\sigma 0} \cdot f_0 + \cdots + e_{\sigma(m-1)} \cdot f_{m-1} + c)$. □

Having proved the two lemmas, we can complete the proof of Theorem 3.3.

PROOF OF THEOREM 3.3. Suppose E is a well-behaved recursive specification. By 3.5, there is a closed $BPA^*_{0,1}$-term for every well-behaved variable of E. Using reasoning as in the lemmas, we can find a closed term for every variable of E. These closed terms constitute a solution for E in $\mathbb{P}$, as $\mathbb{P}$ is a model of $BPA^*_{0,1}$ + RSP*. □

Note 3.6. Our notion of cycling was inspired by, but is different from, the notion of ruling in De Nicola and Labella [2003]; our notion of well behaved was inspired by their notion of hierarchical. It should be noted that they work in a different setting, as they use the law $x \cdot 0 = 0$, which is invalid in the present setting. Another difference is that the law $x + x = x$ is not valid in their setting, but this is not the crucial difference for the present results.

On the other hand, the present work also can be applied in their setting. Adding the law $x \cdot 0 = 0$ amounts to considering the constant 0 as *predictable failure* in the words of Baeten and Bergstra [1990]. In Baeten and Bergstra [1990], it is proven that using this law, every closed term over BPA_0 is either equal to 0 or can be written without 0. This can be extended to $\mathrm{BPA}^*_{0,1}$, since $0^* = 1$, and using a normal form without 0 in the sequel will make all results go through.

4. *Regular Expressions*

Now we consider the reverse direction, how to transform a given regular expression into a well behaved recursive specification of a particular form. Recall from Section 2 that each closed term over $\mathrm{BPA}^*_{0,1}$ can be written as 0, 1 or in the form

$$a_1 \cdot t_1 + \cdots + a_n \cdot t_n + u_1^* \cdot v_1 + \cdots + u_m^* \cdot v_m + \{1\},$$

for certain $n, m \in \mathbb{N}$ with $n + m > 0$, certain $a_i \in A$ and normal forms t_i, u_j, v_j, with $u_j \not\downarrow$. The 1 summand may or may not occur.

Starting from such a normal form, we describe an algorithm to arrive at a recursive specification. Consider an example, taken from De Nicola and Labella [2003]. Take $e = a(a^*b + c) + (c^* + a^*b)^*c^* + a$. Since $c^* \downarrow$, we first need to rewrite this to $a(a^*b + c) + (cc^* + a^*b)^*c^* + a$. Writing this term in normal form, we obtain $e' = a((a1)^*b1 + c1) + (c(c1)^*1 + (a1)^*b1)^*(c1)^*1 + a1$. Associate X_ϵ to e'.

(1) $X_\epsilon = aX_0 + 1X_1 + aX_2$. Thus, X_0 is associated to $(a1)^*b1 + c1$, X_1 to $(c(c1)^*1 + (a1)^*b1)^*(c1)^*1$ and X_2 to 1.
(2) $X_0 = 1X_{00} + cX_{01}$. Thus, X_{00} is associated to $(a1)^*b1$, X_{01} to 1.
(3) $X_{00} = X_{000} + X_{001}$. Each star-term is split into two parts: a part where the loop is executed at least once, and a part where the exit is chosen. Such a term will turn into a cycling variable. Here, X_{000} corresponds to $a1 \cdot X_{00}$, and X_{001} corresponds to $b1$.
(4) $X_{000} = aX_{0000}$.
(5) $X_{0000} = 1X_{00}$. Variable X_{0000} cycles back to X_{00}.
(6) $X_{001} = bX_{0010}$.
(7) $X_{0010} = 1$.
(8) $X_{01} = 1$.
(9) $X_1 = X_{10} + X_{11}$. Again, a star-term is split into two parts. Here, X_{10} corresponds to $(c(c1)^*1 + (a1)^*b1) \cdot X_1$, and X_{11} corresponds to $(c1)^*1$.
(10) $X_{10} = cX_{100} + X_{101}$. Here, X_{100} corresponds to $(c1)^*1 \cdot X_1$, and X_{101} corresponds to $(a1)^*b1 \cdot X_1$.
(11) $X_{100} = X_{1000} + X_{1001}$. Again, a star term.
(12) $X_{1000} = cX_{10000}$.
(13) $X_{10000} = 1X_{100}$. Variable X_{10000} cycles back to X_{100}.
(14) $X_{1001} = 1X_1$. Variable X_{1001} cycles back to X_1.
(15) $X_{101} = X_{1010} + X_{1011}$. Split of star.
(16) $X_{1010} = aX_{10100}$.
(17) $X_{10100} = 1X_{101}$. Variable X_{10100} cycles back to X_{101}.
(18) $X_{1011} = bX_{10110}$.

(19) $X_{10110} = 1X_1$. Variable X_{10110} cycles back to X_1.
(20) $X_{11} = X_{110} + X_{111}$. Split of star.
(21) $X_{110} = cX_{1100}$.
(22) $X_{1100} = 1X_{11}$.
(23) $X_{111} = 1$.
(24) $X_2 = 1$.

Note that the resulting recursive specification is guarded. Furthermore, observe that the resulting specification is much more restricted than the general format of well-behaved specifications. It has two special properties:

(1) Every recursive equation is of the form

$$X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \cdots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\rho \cdot X_\rho + c,$$

where we have that all expressions $e_{\sigma i}$ are constants in $A \cup \{1\}$ so different from 0, and the last two summands are optional and, if present, then $e_\rho, c \in \{0, 1\}$.

(2) If $X_\sigma \searrow X_\tau$, then either X_σ or X_τ is a cycling variable.

Let us call a well-behaved specification satisfying these two properties a well-behaved specification *in restricted form*. If X_σ has a summand of the form $a \cdot X_\tau$, we write $X_\sigma \xrightarrow{a} X_\tau$. The notation $X_\sigma \searrow X_\tau$ used in the previous section now means that X_σ has a summand of the form $1 \cdot X_\tau$.

In general, we define a well-behaved recursive specification in restricted form with solution a given $\mathrm{BPA}^*_{0,1}$-term e by structural induction on e.

PROPOSITION 4.1. *Let e be a closed $BPA^*_{0,1}$-term. There is an effective algorithm giving a well-behaved recursive specification in restricted form with solution e.*

PROOF. The proof goes by structural induction on e, assuming e is given as a normal form as given in Section 2. In the base case, $e \in \{0, 1\}$, and we get the specification $X_\epsilon = e$, so the results are immediate (the variable is not cycling).

In the induction step, we have $e = a_0 \cdot t_0 + \cdots + a_{n-1} \cdot t_{n-1} + u_0^* \cdot v_0 + \cdots + u_{m-1}^* \cdot v_{m-1} + \{1\}$ for certain $n, m \in \mathbb{N}$ with $n + m > 0$, certain $a_i \in A$ and simpler terms t_i, u_j, v_j, with $u_j \not\Downarrow$. By induction hypothesis, we can produce well-behaved recursive specifications E_i, F_j, G_j in restricted form with these terms as solutions. We proceed to define a recursive specification as follows:

(1) $X_\epsilon = a_0 \cdot X_0 + \cdots + a_{n-1} \cdot X_{n-1} + X_n + \cdots + X_{n+m-1} + \{1\}$.
(2) For each $i = 0, \ldots, n-1$, the set of equations E'_i which is produced from E_i by replacing each occurring variable X_σ by $X_{i\sigma}$.
(3) For each $j = 0, \ldots, m-1$, the equation $X_{n+j} = X_{(n+j)0} + X_{(n+j)1}$.
(4) For each $j = 0, \ldots, m-1$, the set of equations F'_j which is produced from F_j by replacing each occurring variable X_σ by $X_{(n+j)0\sigma}$ and replacing each constant summand c by $c \cdot X_{n+j}$.
(5) For each $j = 0, \ldots, m-1$, the set of equations G'_j which is produced from G_j by replacing each occurring variable X_σ by $X_{(n+j)1\sigma}$.

Now fix $j \in \{0, \ldots, m-1\}$, and consider the specification defined for X_{n+j} in the last three items.

First of all, note that this specification is guarded: for all variables $X_{(n+j)1\sigma}$ in G'_j we have $X_{(n+j)1\sigma} \not\searrow X_{(n+j)1\sigma}$ as $X_\sigma \not\searrow X_\sigma$ in G_j; on the other hand, if for some variable $X_{(n+j)0\sigma}$ in F'_j we would have $X_{(n+j)0\sigma} \searrow X_{(n+j)0\sigma}$, this cannot be due to a cycle of 1-steps in F_j by the same argument, so we must have $X_{(n+j)0\sigma} \searrow X_{n+j} \searrow X_{(n+j)0\sigma}$. This implies $X_{(n+j)0} \searrow X_{n+j}$, and in turn that the initial variable of F_j satisfies $\downarrow$, which means $u_j \downarrow$ and this is a contradiction. Finally, the guardedness of $X_{(n+j)1}$ and $X_{(n+j)0}$ imply the guardedness of X_{n+j}.

Next, variable X_{n+j} is a cycling variable: its equation is in the required form, and every exit c in F_j is turned into a term $c \cdot X_{n+j}$ that cycles back. Further, cycling variables in F'_j, G'_j exactly correspond to cycling variables in F_j, G_j (just a prefix added to the index). Thus, the specification of X_{n+j} is in suitable form.

Further, variable X_{n+j} is well behaved: each variable $X_{(n+j)1\sigma}$ is well behaved in G'_j as the corresponding X_σ is well behaved in G_j, and each variable $X_{(n+j)0\sigma}$ cycles back to X_{n+j} as the corresponding X_σ is well behaved in F_j. Taking $\sigma = \epsilon$ yields the well behavedness of X_{n+j}. It is easy to show that the specification is in restricted form.

Finally, using RDP and RSP, from the fact that u_j is a solution of F_j we can infer $X_{(n+j)0} = u_j \cdot X_{n+j}$, and so $X_{n+j} = X_{(n+j)0} + X_{(n+j)1} = u_j \cdot X_{n+j} + v_j = u_j^* \cdot v_j$, where the last step follows from RSP* since $u_j \not\downarrow$.

Now this is established for each $j \in \{0, \ldots, m-1\}$, we can consider the whole specification. Establishing guardedness and preservation of cycling variables is easier than in the previous case, thus the specification is in suitable form. All variables X_i are well behaved, since the initial variables of E_i are well behaved, and so X_ϵ is well behaved. It is easy to show that the specification is in restricted form. Finally, using RDP and RSP, from the fact that t_i is a solution of E_i we can infer $X_i = t_i$, and so $X_\epsilon = a_0 \cdot X_0 + \cdots + a_{n-1} \cdot X_{n-1} + X_n + \cdots + X_{n+m-1} + \{1\} = a_0 \cdot t_0 + \cdots + a_{n-1} \cdot t_{n-1} + u_0^* \cdot v_0 + \cdots + u_{m-1}^* \cdot v_{m-1} + \{1\} = e$. □

Thus, for each closed $\mathrm{BPA}^*_{0,1}$-term we can find a well-behaved recursive specification in restricted form that has this term as a solution.

5. *A Decision Procedure*

Next, we give a decision procedure in order to decide whether a given finite transition system has a well-behaved recursive specification or not. Suppose we have given a finite transition system.

The following proposition is reminiscent of the pumping lemma in formal language theory. It provides a bound on the set of well-behaved recursive specifications we need to consider.

PROPOSITION 5.1. *Let a finite transition system be given with n states and branching degree of k ($k \geq 2$). If this transition system is bisimilar to a well-behaved specification, then it is bisimilar to a restricted well-behaved specification with index set S where all sequences in S have length less than $(n+1)^3 \cdot 2^{3k}$ and entries less than k.*

PROOF. Let a finite transition system be given with n states and branching degree of k that is bisimilar to a well-behaved specification. Due to the results of

the previous two sections, we can take a *restricted* well-behaved specification E that is bisimilar to the given transition system. Each variable in the specification is bisimilar to a state or a substate of the given transition system (if some outgoing transitions of a state are omitted, we get a substate of this state; thus, if $X_\sigma \searrow X_\rho$, then X_ρ corresponds to a substate of the state given by X_σ). A *descending path* in the specification is a sequence of variables $X_\sigma, X_{\sigma i_1}, X_{\sigma i_1 i_2}, \ldots$ such that for each pair of consecutive variables $X_\tau, X_{\tau i}$ we have either $X_\tau \stackrel{a}{\rightarrow} X_{\tau i}$ or $X_\tau \searrow X_{\tau i}$. The proposition follows from the following three key observations.

(1) We can assume that each descending path of length $n+1$ contains a cycling variable.
(2) We can assume that each equation of each variable has at most k summands.
(3) We can assume that each descending path contains at most $n^2 \cdot 2^{3k}$ cycling variables.

For, if we have these three observations, then we can assume that each descending path starts with a series of steps of at most $n+1$ to the first cycling variable, followed by a series of steps of at most $n+1$ to the next cycling variable, and so on, so we have at most $n^2 \cdot 2^{3k} + 1$ blocks between at most $n^2 \cdot 2^{3k}$ cycling variables, and we can limit the length of any descending path to $(n+1)^3 \cdot 2^{3k}$. Thus, the index set of variables only needs to contain sequences of length at most $(n+1)^3 \cdot 2^{3k}$, and the number of summands in any given equation is bounded by k.

It remains to show the three observations. For the first one, suppose there is a descending path of length $n+1$ without a cycling variable. By the restricted format, this means that for each pair of consecutive variables $X_\tau, X_{\tau i}$ there must be some atomic action a such that $X_\tau \stackrel{a}{\rightarrow} X_{\tau i}$, and each variable in the descending path, except maybe the first one, corresponds to a state in the given transition system. As a result, two distinct variables in this path, say X_σ and $X_{\sigma\rho}$, must correspond to the same state in the given transition system. Now consider the specification where the part below X_σ is replaced by the part below $X_{\sigma\rho}$, that is, we throw out all equations of variables of the form $X_{\sigma\pi}$, and we put in new equations

$$X_{\sigma\pi} = c_0 \cdot X_{\sigma\pi 0} + \cdots + c_{m-1} \cdot X_{\sigma\pi(m-1)} + c_m \cdot X_\xi + c_{m+1}$$

whenever there was an equation

$$X_{\sigma\rho\pi} = c_0 \cdot X_{\sigma\rho\pi 0} + \cdots + c_{m-1} \cdot X_{\sigma\rho\pi(m-1)} + c_m \cdot X_{\xi'} + c_{m+1}$$

skipping the ρ part in the summands. If an occurring cycling variable X_ξ has a $\sigma\rho$ prefix, then also there the ρ part can be skipped; otherwise, it must lie before X_σ, and can remain unchanged. The resulting specification is again well-behaved, and in restricted form, because E is in restricted form and $X_\sigma, X_{\sigma\rho}$ are not cycling variables. This procedure can be repeated until the specification has no descending paths of length $n+1$ without a cycling variable.

For the second observation, suppose there is a variable X_σ whose equation contains more than k summands. In this case, X_σ must be bisimilar to a state or a substate s of the given transition system. A substate of a state is obtained by leaving out some outgoing transitions. This (sub)state has a number of transitions $s \stackrel{a}{\rightarrow} s'$, and maybe a termination option $s \downarrow$, numbering in total at most k. By bisimulation,

each of these transitions or termination option must be matched by at least one of the summands of X_σ. For each of them, pick one of the summands where it is matched, in total at most k. Now all other summands can be left out (together with their entire subspecifications), resulting in an equivalent specification. Next, some renaming is required to obtain again a downwards closed index set. Due to the fact that in this simplification step cycling variables are only removed together with incoming transitions as well as with their entire subspecifications, the resulting specification is again in restricted form.

For the third observation, first we do both reductions of the two previous cases, so we can assume that each descending path of length $n + 1$ contains a cycling variable, and each variable has at most k summands.

CLAIM 5.2. *Cycling variables can be nested only $n \cdot 2^{2k}$ deep, that is, there are at most $n \cdot 2^{2k}$ cycling variables where each variable is in the cycling part of the previous one.*

In order to prove the claim, suppose not. Each cycling variable is bisimilar to a state or a substate of the given transition system. As this transition system has at most n states, and a branching degree at most k, it has at most $n \cdot 2^k$ substates. If there are more than $n \cdot 2^k$ nested cycling variables, there must be two that are bisimilar, so there are X_σ, $X_{\sigma\rho}$ such that $X_{\sigma\rho}$ is in the cycling part of X_σ, that is, it is below $X_{\sigma 0}$. Now X_σ, $X_{\sigma\rho}$ are bisimilar, but it need not be the case that $X_{\sigma 0}$, $X_{\sigma\rho 0}$ are bisimilar, as the split into the cycling and the exit part can be done differently in the two cases. But notice that there are only 2^k different cycling parts possible, as each outgoing transition will belong to the cycling part or not. Thus, if there are more than $n \cdot 2^{2k}$ nested cycling variables, there must be two that are bisimilar and that moreover have bisimilar cycling parts. Thus, it must be the case that there are X_σ, $X_{\sigma\rho}$ such that $X_{\sigma\rho}$ is below $X_{\sigma 0}$, and X_σ, $X_{\sigma\rho}$ are bisimilar, and moreover $X_{\sigma 0}$, $X_{\sigma\rho 0}$ are bisimilar.

Now consider the specification where the part from $X_{\sigma 0}$ is replaced by the part from $X_{\sigma\rho 0}$, that is, we replace the cycling part of the cycling variable X_σ and keep the exit part (the part from $X_{\sigma 1}$). This replacement is done in the same way as outlined in the proof of the first observation, skipping the ρ part in the cycling variables. The result is a well-behaved specification in restricted form that has fewer cycling variables and still is bisimilar to the given transition system. This means we have proved the claim.

Next, notice that we can assume that there are at most $n \cdot 2^k$ consecutive cycling variables on a descending path such that each cycling variable is in the exit part of the previous one. For, if not, then there must be two cycling variables that correspond to the same substate of the given transition system, and we can replace the first one by the second one, resulting in a bisimilar well-behaved specification in restricted form.

Now, given these observations, it can still happen that there is a sequence of cycling variables, that alternatingly occur in the cycling part and in the exit part. This means there can be cycling variables X_σ, $X_{\sigma\rho}$, $X_{\sigma\rho\pi}$ such that X_σ is bisimilar to $X_{\sigma\rho\pi}$, $X_{\sigma\rho}$ is below $X_{\sigma 1}$ (the exit part of X_σ) and $X_{\sigma\rho\pi}$ is below $X_{\sigma\rho 0}$ (the cycling part of $X_{\sigma\rho}$). To illustrate this phenomenon, we give an example. Consider the regular expression $ab^*c(db^*c)^*e$. This expression gives rise to the following well-behaved recursive specification (omitting all the extra 1's induced by the normal

form):

$$\begin{aligned}
X_\lambda &= aX_0\\
X_0 &= X_{00} + X_{01}\\
X_{00} &= bX_0\\
X_{01} &= cX_{010}\\
X_{010} &= X_{0100} + X_{0101}\\
X_{0100} &= dX_{01000}\\
X_{01000} &= X_{010000} + X_{010001}\\
X_{010000} &= bX_{01000}\\
X_{010001} &= cX_{010}\\
X_{0101} &= e.
\end{aligned}$$

Now cycling variables X_0 and X_{01000} correspond to bisimilar states of the process. The second one occurs inside the cycling part of variable X_{010}, the first one does not. In the case of this specification, it turns out that it cannot be reduced to a simpler well-behaved specification.

However, combining the last observation with the claim, we see that the total number of cycling variables on a descending path is bounded by $n^2 \cdot 2^{3k}$. For, the total number of nested cycling variables is at most $n \cdot 2^{2k}$, and between one of these and the following, there can be at most $n \cdot 2^k$ cycling variables each of which is in the exit part of the previous one. □

This proposition gives a bound on the size of the specification that we need to consider. We expect that this bound can be tightened further, in fact we have a further reduction that reduces $(n+1)^3 \cdot 2^{3k}$ to $(n+1)^3 \cdot 2^{2k}$. This bound immediately gives rise to a decision procedure, as there are only finitely many regular recursive specifications within the bound. We can check for each one, whether or not it is bisimilar to the given transition system, as bisimulation is decidable on finite transition systems.

To give an example of a transformation into well-behaved form, consider the guarded recursive specification $\{X = aY, Y = bX + aZ, Z = cX + aY\}$. In this form, it is not a well-behaved recursive specification. It turns into one, by replacing X by aY everywhere on the right-hand side. We get the following specification:

$$\begin{aligned}
X_\lambda &= aX_0\\
X_0 &= X_{00} + X_{01}\\
X_{00} &= bX_{000} + aX_{001}\\
X_{000} &= aX_0\\
X_{001} &= cX_{0010} + aX_0\\
X_{0010} &= aX_0\\
X_{01} &= 0.
\end{aligned}$$

6. *Star Height*

In this section, we give two applications of the concept of a well-behaved specification, of the results linking these specifications with the process interpretation, and of the methods used for obtaining the decidability result in the previous section. In particular, we consider two questions involving the minimal star height of regular expressions under bisimulation. We solve the star height problem for the process

interpretation, and give an alternative proof of a result by Hirshfeld and Moller [1996] concerning the star-height question posed by Milner [1984].

In formal language theory, the determination of the star height of a regular language is an old problem, which originated with Eggan [1963]. The *star height* of a regular expression e is defined syntactically as the maximum number of nested stars that e contains.

Definition 6.1. Let t be a closed $\mathrm{BPA}^*_{0,1}$-term. The *star height* of t, $sh(t)$ is defined inductively:

(1) $sh(0) = sh(1) = sh(a) = 0$ (for all actions $a \in A$);
(2) $sh(t + s) = sh(t \cdot s) = \max\{sh(t), sh(s)\}$;
(3) $sh(t^*) = 1 + sh(t)$.

In language theory, the interest is in the star height of regular languages: the *star height* of a regular language L, $sh(L)$, is the least natural number n such that $sh(e) = n$ for some regular expression e that represents L. Again, it has to be noted that complementation and intersection are not included as regular operators. When these are included, we talk about generalized star height.

Many results concerning the star height of regular languages are known (still only much less is known about generalized star height, but see, e.g., Pin et al. [2001]), of which we only mention three:

(1) Every regular language over a single-letter alphabet is of star height at most one.
(2) There are regular languages with any preassiged star height over any alphabet containing at least two letters [McNaughton 1966].
(3) There exists an algorithm for computing the star height of a regular language given by a regular expression [Hashiguchi 1983]; this famous statement established that the star-height problem in formal language theory is solvable.

Here we are interested in counterparts for regular expressions under the process interpretation of these star-height results for regular languages. For this purpose, we define, in analogy to the star height of a regular language, the minimal star height of a regular expression under the process interpretation.

Definition 6.2. The *minimal star height* of a regular expression e (under the process interpretation) is the least natural number n such that there exists a regular expression f with $sh(f) = n$ and $f \leftrightarrows e$.

We start by investigating a counterpart for the star-height problem under the process interpretation. Thus, this is the problem of determining the minimal star height of a given regular expression under the process interpretation.

In analogy to the result of Hashiguchi [1983], we show that this problem is algorithmically solvable, making use of our results concerning the relationship between regular expressions under bisimulation and well-behaved specifications, and of our decidability result in the previous section.

Given the correspondence between regular expressions and well-behaved specifications proved earlier, star height can also be defined for well-behaved specifications.

Definition 6.3. Let E be a recursive specification over $\mathrm{BPA}^*_{0,1}$ in suitable form over variables $\{X_\sigma : \sigma \in S \subset \mathbb{N}^*\}$. Define, for each variable X_σ, its *star height* $sh(X_\sigma)$ by induction on the variable tree below X_σ:

—If X_σ is a cycling variable, define $sh(X_\sigma) = \max\{1 + sh(X_{\sigma 0}), sh(X_{\sigma 1})\}$;
—If X_σ is not a cycling variable, its equation is of the form

$$X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \cdots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho + c,$$

and we define

$$sh(X_\sigma) = \max\{sh(e_{\sigma 0}), sh(X_{\sigma 0}), \ldots, sh(e_{\sigma(m-1)}), sh(X_{\sigma(m-1)}), sh(e^*_\sigma), sh(e_\rho)\}.$$

Finally, the star height of E is defined by $sh(E) = sh(X_\epsilon)$.

Now Lemmas 3.4 and 3.5 can be strengthened to obtain the following two lemmas, which can be proved by following step by step the earlier proofs, using the definition just given. The first lemma is used in the proof of the second.

LEMMA 6.4. *Let E be a recursive specification in suitable form, and suppose X_σ cycles back to X_ρ. Then, there is a closed term e over $BPA^*_{0,1}$ with $sh(e) = sh(X_\sigma)$ such that $X_\sigma = e \cdot X_\rho$.*

LEMMA 6.5. *Let E be a recursive specification in suitable form, and suppose variable X_σ is well-behaved. Then, there is a closed term e over $BPA^*_{0,1}$ with $sh(e) = sh(X_\sigma)$ such that $X_\sigma = e$.*

As an immediate consequence of the second lemma and of its proof as hinted above, the next lemma states that, for every well-behaved recursive specification E, a solution process in the form of a closed term of the same star height as E can be found algorithmically.

LEMMA 6.6. *There is a procedure that on the input of a well-behaved recursive specification E over $BPA^*_{0,1}$ outputs a closed term e over $BPA^*_{0,1}$ such that e is a solution of E and $sh(e) = sh(E)$ holds.*

Conversely, the procedure of Section 4 assigns to each regular expression a well-behaved recursive specification of the same star height. This is the statement of the following lemma. The result can again be found by following the procedure step by step, checking that star height is preserved at every step.

LEMMA 6.7. *There is a procedure that on the input of a closed term e over $BPA^*_{0,1}$ outputs a well-behaved specification E in restricted form with e as a solution that satisfies $sh(E) = sh(e)$.*

Based on the last two lemmas, the star-height problem can be solved.

THEOREM 6.8. *The star-height problem in $BPA^*_{0,1}$ is solvable: There is an algorithm that, for all closed terms e over $BPA^*_{0,1}$, on input e computes the minimal star height of e.*

PROOF. We describe an algorithm for computing, on the input of a closed term e, its minimal star height m.

(1) With the procedure described in Section 4, produce a well-behaved recursive specification E that has e as a solution. Then, $sh(E) = sh(e)$ holds due to Lemma 6.7.

(2) Using the operational rules of Table II, produce the finite transition system of e. Let n be the number of states of this transition system, and k its branching degree. Define $N = (n+1)^3 \cdot 2^{3k}$.

(3) Generate successively the finite number of well-behaved specifications over $\mathrm{BPA}^*_{0,1}$ in restricted form with depth less than N and with branching degree less than or equal to k. For each of these recursive specifications E, check whether E has e as a solution, and if so, then compute the star height $sh(E)$ of E. In the search through this finite number of specifications determine the minimum m of the star heights of specifications E generated that have e as a solution. The number m is well defined here because by applying Proposition 5.1 to the specification E obtained in the first step there exists at least one well-behaved specification with the properties desired.

Finally, output m as the minimal star height of e.

In order to show that this output m is really the minimal star height of e, we first observe that the minimal star height of e is less than or equal to m: by the definition of m, there exists a well-behaved specification D in restricted form of star height m with e as solution. By Lemma 6.6, there exists a solution d of D with $sh(d) = sh(D) = m$. Since $d \leftrightarrows e$, the minimal star height of e is at most m.

Now it remains to show that the minimal star height of e is at least m. For this, suppose f is an arbitrary closed term over $\mathrm{BPA}^*_{0,1}$ that is bisimilar to e. We have to show $sh(f) \geq m$.

By Lemma 6.7, there exists a well-behaved specification F in restricted form with f as a solution such that $sh(F) = sh(f)$. By using the simplifications to well-behaved specifications described in the proof of Proposition 5.1, from F a well-behaved specification F' can be found that is equivalent to F, has depth less than N and branching degree less than or equal to k. Furthermore, $sh(F') \leq sh(F)$ holds, because the simplifications used in the proof of Proposition 5.1 do not increase (but may decrease) the star height of the specification. It follows that $sh(F') \leq sh(f)$.

Term e is also a solution of F', because e is bisimilar to f, F' is equivalent to F, and f is a solution of F. Due to this and the fact that the depth of F' is less than N and that its branching degree is less than or equal to k, the specification F' is generated in the third step of the algorithm on input e, and F' is taken into account when determining the minimum m. It follows that $sh(F') \geq m$, and so $sh(f) \geq m$. □

Regarding a possible counterpart of the first statement of the star height of regular languages over a singleton alphabet, Milner [1984] conjectured that the predicate "minimal star height is less or equal to n" defines a proper hierarchy on the set of regular expressions even over singleton alphabets under bisimulation. In particular, he suggested the sequence $\{f_n\}$ of regular expressions defined inductively by

$$f_1 = a^* , \qquad f_{n+1} = (f_n \cdot a)^* \quad (\text{for all } n > 0) , \tag{1}$$

and conjectured that the minimal star height of f_n is actually n. Hirshfeld and Moller [2000] showed that this is indeed the case.

THEOREM 6.9. [HIRSHFELD AND MOLLER 2000]. *For every $n > 0$, there exists a regular expression f_n over the single-letter alphabet $\{a\}$ such that the minimal star height of f_n under the process interpretation is n. This is witnessed by the sequence $\{f_n\}_n$ that is defined inductively by* (1).

As a consequence of this theorem, the predicate "minimal star height is less or equal to n", where n is a natural number, defines a proper hierarchy on the set of regular expressions over indeed every non-empty alphabet. This contrasts with the first star-height result for regular languages, while not with the second result, for the language interpretation of regular expressions.

It is easy to recognise that in the case of empty alphabets an analogous result as for regular languages holds for the minimal star height under the process interpretation: if the set of atomic actions is empty, then each closed term is bisimilar to one of the constants 0 or 1, and hence all closed terms have minimal star height 0. This corresponds with the fact that the regular language over the empty alphabet has star height zero.

We will give an alternative proof for Theorem 6.9. For this, we first need to develop a number of auxiliary statements.

For all $n > 0$, we denote by F^n the regular recursive specification that contains precisely the following equations:

$$\begin{aligned} Y_n &= a \cdot Y_1 + \cdots + a \cdot Y_n + 1 \\ Y_{n-1} &= a \cdot Y_1 + \cdots + a \cdot Y_n \\ &\cdots \\ Y_1 &= a \cdot Y_1 + a \cdot Y_2. \end{aligned}$$

PROPOSITION 6.10. *Consider the recursive specification F^n. Then $Y_i \leftrightarrows Y_j \iff i = j$ for all $i, j \in \{1, \ldots, n\}$.*

PROOF. The proposition is an easy consequence of the fact that, for all $i \leq n$, the shortest execution trace of Y_i that leads to termination consists of $n - i$ transitions. □

Note that for $i < j$, Y_i is bisimilar to a substate of Y_j. We can express this as shown in the following lemma.

PROPOSITION 6.11. *Consider again specification F^n. Then $Y_i + Y_j \leftrightarrows Y_j \iff i \leq j$, for all $i, j \in \{1, \ldots, n\}$.*

The following lemma relates the closed terms of the family $\{f_n\}$ defined above with each other via derivability.

LEMMA 6.12. *For all $n > 0$ the following equation is derivable from $BPA^*_{0,1} + RSP^*$:*

$$f_n = af_1 \cdots af_n + af_2 \cdots af_n + \cdots + af_{n-1}af_n + af_n + 1.$$

PROOF. A straightforward induction on n. □

Based on this lemma, the next lemma relates the closed terms f_n with the recursive specifications F^n introduced above.

LEMMA 6.13. *For all $n > 0$, f_n is a solution of the recursive specification F^n.*

PROOF SKETCH. The sequence of terms $f_1af_2a\cdots f_{n-1}af_n$, $f_2a\cdots f_{n-1}af_n, \ldots, f_{n-1}af_n$, f_n can be substituted for the recursion variables $Y_1, Y_2, \ldots, Y_{n-1}, Y_n$, respectively. □

In order to determine the minimal star height of f_n we need to consider arbitrary well-behaved specifications E that have f_n as a solution, and hence specifications that are equivalent to the specification F^n defined above. The following lemma states that for a well-behaved specification E in restricted form equivalent to F^n, and a variable X_σ in E, this variable must be bisimilar to a state Y_i, or to a substate of Y_i, for a variable Y_i of F^n.

LEMMA 6.14. *Let $n > 0$, and let E be a well-behaved specification in restricted form such that E is equivalent to F^n. Then, for each X_σ in E, there is $i \leq n$ such that $X_\sigma + Y_i \leftrightarrows Y_i$.*

Thus, each variable X_σ is bisimilar to a substate of some Y_i. We will need the *least* such i for each σ, so we define

$$i_\sigma = \min\{i \in \{1, \ldots, n\} \mid X_\sigma + Y_i \leftrightarrows Y_i\}.$$

The next lemma now states a key property. When a variable cycles back to another variable, then this least i increases, unless there is a path consisting only of 1-steps.

LEMMA 6.15. *Let E be a well-behaved specification in restricted form that is equivalent to the regular specification F^n. Then for all σ, ρ such that X_σ cycles back to X_ρ one of the following two statements holds: either $i_\sigma < i_\rho$, or $i_\sigma = i_\rho$ and $X_\sigma \searrow X_\rho$.*

PROOF HINT. By induction on $n - i_\sigma$. □

Now we get to the key lemma needed in the proof of the following theorem.

LEMMA 6.16. *Let E be a well-behaved specification in restricted form equivalent to F^n. Let σ be such that X_σ is well-behaved, or, for some ρ, X_σ cycles back to X_ρ and $X_\sigma \not\searrow X_\rho$. Furthermore, let $i > 0, i \leq n$ be such that either $X_\sigma \leftrightarrows Y_i$, or there exists a ξ such that $X_{\sigma\xi}$ cycles back to X_σ, $X_{\sigma\xi} \searrow X_\sigma$ and $X_{\sigma\xi} \leftrightarrows Y_i$. Then $sh(X_\sigma) \geq i$ holds.*

PROOF HINT. The lemma can be shown by induction on i, with a subinduction on the depth of the variable tree below X_σ in E. The induction step is a careful, but tedious analysis of a number of possible cases. A crucial tool is Lemma 6.15. □

This key lemma now enables us to show that every well-behaved specification in restricted form that is equivalent to the regular specification F^n must have star height greater than or equal to n.

THEOREM 6.17. *Let E be a well-behaved specification in restricted form that is equivalent to the specification F^n, for some $n > 0$. Then, $sh(E) \geq n$.*

PROOF. The theorem is an immediate consequence of Lemma 6.16 in view of the fact that the leading variable X_ϵ of a well-behaved specification E is well behaved. □

Now we are finally in a position to give our proof for Theorem 6.9.

PROOF OF THEOREM 6.9. We have to show that, for all $n > 0$, the minimal star height of f_n is n.

We first notice that $sh(f_n) = n$ holds for all $n > 0$: this follows by straightforward induction on n using the inductive definition of the closed terms f_n. As a consequence, the minimal star height of f_n is at most n.

Therefore, it remains to show that the minimal star height of f_n is at least n. For this, take $n > 0$ and let e be an arbitrary closed term with $e \leftrightarrows f_n$. We aim to show $sh(e) \geq n$.

Starting from the closed term e, we build, by using the procedure guaranteed by Lemma 6.7, a well-behaved specification E in restricted form that has e as a solution. Due to Lemma 6.7 we find $sh(E) = sh(e)$. Since $e \leftrightarrows f_n$, since e is a solution of E, and f_n is, due to Lemma 6.13, a solution of the regular specification F^n, it follows that the specifications E and F^n are equivalent, that is, they have the same solutions. Thus $X_\epsilon \leftrightarrows Y_n$. By Theorem 6.17, this implies $sh(e) \geq n$.

In this way, we have shown, for arbitrary $e \leftrightarrows f_n$ that $sh(e) \geq sh(f_n) = n$. Hence, the minimal star height of the term f_n, for all $n > 0$, is at least n. It follows that the closed terms f_n have minimal star height n. □

In this proof, we have made use of the structure of well-behaved specifications to show that if $e \leftrightarrows f_n$ (entailing that e is a solution of the regular specification F^n) and e is a solution of a well-behaved specification E in restricted form, then $sh(E) \geq n$. From this we concluded that the minimal star height of f_n is greater or equal to n, and finally, that it is equal to n. In contrast to this, Hirshfeld and Moller in their proof of Theorem 6.9 reason about "decorated regular expressions" containing specially bracketted subterms which allow to get insight into the star nesting structure of a closed term e that is a solution to a specification F^n in the course of infinite execution paths from e.

Apart from this conceptual aspect, another difference consists in the fact that our proof, and in particular the proof of Lemma 6.16, proceeds by an induction in direction bottom-up, starting close to the bottom of the specification considered, whereas the proof in Hirshfeld and Moller [2000] works essentially top-down. In particular, Hirshfeld and Moller [2000] show, by induction on j, that the following statement holds for all $n > 0$:

> If $e \leftrightarrows f_n$ holds for a closed term e, then there exist, for all $j \in \{1, \ldots, n-1\}$, terms e_j such that e can evolve to e_j in a number of steps such that $e_j \leftrightarrows Y_{n-j}$, and such that $sh(e_j) \geq j + 1$.

By letting $j = n - 1$, and by using the fact that transitions do not increase (but may decrease) the star height of terms, it then follows that all closed terms e that are bisimilar to f_n have star height at least n. In comparison to this proof, it can be shown that Lemma 6.16 entails the rather similar, but different, statement saying it holds for all $n > 0$:

> If $e \leftrightarrows f_n$ holds for a closed term e, then there exist, for all $i \in \{1, \ldots, n\}$, terms e_i such that e evolves to e_i in a number of steps and $e_i \leftrightarrows Y_{n-i+1}$ and $sh(e_i) \geq n - i + 1$.

Following the induction in the proof of Lemma 6.16, it can be said that this statement is established by induction on $n - i + 1$. By letting $i = 1$, also this statement implies that all closed terms e that are bisimilar to f_n have star height greater or equal to n.

It is possible to cast the proof of Hirshfeld and Moller [2000] as an argument about well-behaved specifications in restricted form, using a top-down induction. We have not done this here, for two reasons: first, the definition of well-behaved specifications and all proofs here have been carried out in a bottom-up manner, starting with recursion variables at the bottom of the specification; and second, we arrived at our proof in the way described here.

7. *Conclusion*

We have defined a set of well-behaved recursive specifications that corresponds exactly to the set of regular expressions, using bisimulation as the notion of equivalence. The same result holds if we restrict to the set of well-behaved recursive specifications in restricted form, that have a rather direct interpretation as a set of finite transition systems. Thus, we can say that we have defined a structural property that characterizes the set of finite automata which are expressible by a regular expression (modulo bisimulation). This means we have solved the open question of Milner [1984].

Given a finite transition system, we have presented a decision procedure to determine whether or not this transition system is equivalent to a well-behaved specification. This decision procedure may still require a large number of specifications to be checked. We are working on a more direct algorithm that will construct a regular expression if one exists. Note that Bosscher [1997] describes an algorithm that decides the analogous problem in the absence of the constant 0, and another algorithm in the absence of the two constants $0, 1$. An obvious question that remains is the following. What is the precise complexity of the decision algorithm, described in the proof of Proposition 5.1, for $\mathrm{BPA}^*_{0,1}$-definability of finite transition systems? Is it possible to implement this procedure as a back-tracking algorithm that is able to decide $\mathrm{BPA}^*_{0,1}$-definability of "small" finite transition systems? Are substantial improvements of this algorithm possible?

Our results can be adapted to the setting of De Nicola and Labella [2003], where the constant 0 really acts as the zero process.

The way Milner [1984] put the open questions suggests that he expected that the solution of the problem we solved here should also lead to a complete axiomatization of regular expresssions under the process interpretation. In fact, we have tried to show that the presented axiomatization of $\mathrm{BPA}^*_{0,1}$ together with the principle RSP* is complete, but have not succeeded as of yet.

As an application of our results, we give an algorithm to determine the star height of a regular expression (under bisimulation). We give an alternative proof of the fact that the star height hierarchy is nontrivial, even in the case of a singleton alphabet.

We find that the expressive power of iteration in concurrency theory is not large, as not all regular processes can be written as a star term. Still, there are specification languages and programming languages that contain both parallel composition and iteration (e.g., van Beek et al. [2006]). More investigation is needed to see if there are operators that also capture an intuitive notion, but have more expressive power. Some investigations are reported in Bergstra et al. [2001].

The notion of a regular process in concurrency theory is clear: it can be given by a finite transition system, and by a finite regular recursive specification. It remains

to capture this in terms of an extra operator, or several extra operators, on top of $\mathrm{BPA}^*_{0,1}$. The notion of a context-free process is not so well developed: it can be given by a finite recursive specification over $\mathrm{BPA}^*_{0,1}$, but we do not have a characterization in terms of transition systems, or in terms of algebraic operators. This is a matter of future research.

The same questions can be asked if we add additional operators: foremost among these is parallel composition, with or without communication. Some results can already be found in Baeten and Bergstra [1988], but many questions remain. Let us denote the theory obtained from $\mathrm{BPA}^*_{0,1}$ by adding parallel composition as $\mathrm{PA}^*_{0,1}$. It is easy to see that, like $\mathrm{BPA}^*_{0,1}$ definable processes, $\mathrm{PA}^*_{0,1}$ definable processes are bisimilar to finite-state processes. In the very similar situation of the theories BPA^*_0 and PA^*_0 (where a binary variant of iteration should be used), it is shown in Bergstra et al. [1994] that the expressivity of PA^*_0 is strictly between that of BPA^*_0 and that of the finite state processes. In particular, it is shown there that the PA^*_0-process $((ab)^*c) \parallel d$ cannot be expressed in BPA^*_0. Using the characterisation, proved in this paper, of $\mathrm{BPA}^*_{0,1}$-definable processes as processes that are solutions of well-behaved specifications, it is not difficult to show that this process is not definable in $\mathrm{BPA}^*_{0,1}$ either. It follows that the expressivity of $\mathrm{PA}^*_{0,1}$ is strictly greater than that of $\mathrm{BPA}^*_{0,1}$. We conjecture that the expressivity of $\mathrm{PA}^*_{0,1}$ is also strictly less than that of the finite-state processes. But this raises analogous questions, now for the slightly more expressive system $\mathrm{PA}^*_{0,1}$, to those that have been solved in this article, namely what structural property characterises the $\mathrm{PA}^*_{0,1}$-definable processes? Is there a decision method for the problem of recognising those finite transition systems that are the process interpretations of closed $\mathrm{PA}^*_{0,1}$-terms?

We can also have a look at extra operators that are used in language theory, such as complementation and intersection, and try to interpret them in the process setting. Complementation should be done with respect to a kind of universal process, that can show all possible behavior. A candidate for such a process is the *Chaos* process of CSP [Hoare 1985].

ACKNOWLEDGMENTS. We thank Colin Stirling (University of Edinburgh), Bas Luttik (Technische Universiteit Eindhoven), Wan Fokkink (Vrije Universiteit, Amsterdam), Luca Aceto (Reykjavik University), Alban Ponse (University of Amsterdam) and the anonymous referees for their useful remarks and suggestions.

REFERENCES

ACETO, L., FOKKINK, W., AND VERHOEF, C. 2001. Structural operational semantics. In *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. Elsevier Science, Amsterdam, The Netherlands, 197–292.

BAETEN, J., AND BERGSTRA, J. 1988. Global renaming operators in concrete process algebra. *Inf. Comput. 78*, 205–245.

BAETEN, J., AND BERGSTRA, J. 1990. Process algebra with a zero object. In *Proceedings CONCUR'90*, J. Baeten and J. Klop, Eds. Lecture Notes in Computer Science, vol. 458. Springer-Verlag, New York, 83–98.

BAETEN, J., BERGSTRA, J., AND KLOP, J. 1987. On the consistency of Koomen's fair abstraction rule. *Theoret. Comput. Sci. 51,* 1/2, 129–176.

BAETEN, J., BERGSTRA, J., AND KLOP, J. 1993. Decidability of bisimulation equivalence for processes generating context-free languages. *J. ACM 40,* 3, 653–682.

BAETEN, J., AND CORRADINI, F. 2005. Regular expressions in process algebra. In *Proceedings LICS'05*. IEEE Computer Society Press, Los Alamitos, CA, 12–19. (Also report CS-R 05-11, Computer Science, Technische Universiteit Eindhoven.)

BAETEN, J., AND VERHOEF, C. 1995. Concrete process algebra. In *Handbook of Logic in Computer Science*, S. Abramsky, D. Gabbay, and T. Maibaum, Eds. Vol. 4. Oxford University Press, Oxford, UK, 149–269.

BAETEN, J., AND WEIJLAND, W. 1990. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, MA.

BERGSTRA, J., BETHKE, I., AND PONSE, A. 1994. Process algebra with iteration and nesting. *Comput. J. 37,* 4, 243–258.

BERGSTRA, J., FOKKINK, W., AND PONSE, A. 2001. Process algebra with recursive operations. In *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. Elsevier Science, Amsterdam, The Netherlands, 333–390.

BERGSTRA, J., AND KLOP, J. 1984. The algebra of recursively defined processes and the algebra of regular processes. In *Proceedings 11th ICALP*, J. Paredaens, Ed. Lecture Notes in Computer Science, vol. 172. Springer-Verlag, New York, 82–95.

BOSSCHER, D. 1997. Grammars modulo bisimulation. Ph.D. University of Amsterdam, Amsterdam, The Netherlands.

CORRADINI, F. 2000. A step forward towards equational axiomatizations of Milner bisimulation in Kleene star. In *Proceedings FICS 2000*.

CORRADINI, F., DE NICOLA, R., AND LABELLA, A. 2002. An equational axiomatization of bisimulation over regular expressions. *J. Logic Comput. 12*, 301–320.

DE NICOLA, R., AND LABELLA, A. 2003. Nondeterministic regular expressions as solutions of equational systems. *Theoret. Comput. Sci. 203*, 179–189.

EGGAN, L. 1963. Transition graphs and the star-height of regular events. *Mich. Math. J. 10*, 385–397.

HASHIGUCHI, K. 1983. Representation theorems on regular languages. *J. Comput. Syst. Sci. 27*, 101–105.

HIRSHFELD, Y., AND MOLLER, F. 1996. Decidability results in automata and process theory. In *Proceedings of Logics for Concurrency: Automata versus Structure. The VIII Banff Higher Order Workshop*, G. Birtwistle and F. Moller, Eds. Lecture Notes in Computer Science, vol. 1043. Springer-Verlag, New York, 102–148.

HIRSHFELD, Y., AND MOLLER, F. 2000. On the star height of unary regular behaviours. In *Proof, Language, and Interaction (Essays in Honour of Robin Milner)*, G. Plotkin, C. Stirling, and M. Tofte, Eds. Foundations of Computing. MIT Press, Cambridge, MA, Chapter 16, 497–509.

HOARE, C. 1985. *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs, NJ.

MCNAUGHTON, R. 1966. The loop complexity of regular events. Tech. Rep., Machine Structures Group Memo 18, MIT, Cambridge, MA.

MILNER, R. 1984. A complete inference system for a class of regular behaviours. *J. Comput. Syst. Sci. 28,* 3, 439–466.

MILNER, R. 1989. *Communication and Concurrency*. Prentice Hall, Englewood Cliffs, NJ.

PIN, J.-E., STRAUBING, H., AND THÉRIEN, D. 2001. Some results on the generalized star-height problem. *Inf. Comput. 101,* 2, 219–250.

PLOTKIN, G. 2004. A structural approach to operational semantics. *J. Logic Algeb. Prog. 60*, 17–139. (Reprint from 1981 in Special Issue on Structural Operational Semantics.)

SÉNIZERGUES, G. 2001. L(A)=L(B)? decidability results from complete formal systems. *Theoret. Comput. Sci. 251*, 1–166.

SEWELL, P. 1997. Nonaxiomatisability of equivalences over finite state processes. *Ann. Pure Appl. Logic 90*, 163–191.

STIRLING, C. 2001. Decidability of DPDA equivalence. *Theoret. Comput. Sci. 255*, 1–31.

TERESE. 2003. *Term Rewriting Systems*. Number 55 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

TROEGER, D. 1993. Step bisimulation is pomset equivalence on a parallel language without explicit internal choice. *Math. Struct. Comput. Sci. 3,* 1, 25–62.

VAN BEEK, D., MAN, K., RENIERS, M., ROODA, J., AND SCHIFFELERS, R. 2006. Syntax and consistent equation semantics of hybrid chi. *J. Logic Algeb. Prog. 68,* 1/2, 129–210.

VAN GLABBEEK, R. 2001. The linear time—Branching time spectrum I. The semantics of concrete, sequential processes. In *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. Elsevier Science, Amsterdam, The Netherlands, 3–100.

RECEIVED OCTOBER 2005; REVISED JUNE 2006; ACCEPTED JANUARY 2007