A Characterization of Regular Expressions under Bisimulation

J.C.M. Baeten
Divison of Computer Science, Technische Universiteit Eindhoven
josb@win.tue.nl

F. Corradini
Department of Mathematics and Computer Science, Università di Camerino
flavio.corradini@unicam.it

C.A. Grabmayer*
Department of Computer Science, Vrije Universiteit, Amsterdam
clemens@cs.vu.nl

Abstract

We solve an open question of Milner ([15]). 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.

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, 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 have a place in every undergraduate curriculum, 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 paper, besides solving a long-standing open problem, opens the way to a deeper study of the relationships between automata theory and concurrency theory.

In formal language theory, there is a well-known correspondence between the set of regular expressions and the set of finite (non-deterministic) automata: for each regular expression a finite automaton can be found that admits the same language, and vice versa. But it is also well-known that this correspondence breaks down if we consider other notions of equivalence, other than language equivalence. Of particular interest is bisimulation equivalence. Milner proves in [15] that not every finite automaton is bisimulation equivalent to a regular expression, a closed term in the process algebra with atomic actions, successful and unsuccessful termination, choice, sequential composition and iteration. He poses the question how the set of finite behaviours that are bisimulation equivalent to a regular expression can be characterized. Here, we solve this open question. We define a set of so-called *well-behaved* finite behaviours, that corresponds exactly to the set of regular expressions. We show how to determine whether a given finite behaviour is well-behaved. As an application, we show how to determine the minimal star height (with respect to bisimulation) of a regular expression. This paper extends [2], where well-behaved specifications were introduced. For other related work, see [10].

1.1 Acknowledgements

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

*Supported by the Netherlands Organisation for Scientific Research NWO in project 612.000.313, GeoProc.

2 Process Algebra

We start out from the equational theory 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. BPA$^*_{0,1}$ extends the basic process algebra BPA (see [6]) 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. 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 [16]). As a result, we have a difference between successful termination (1) and unsuccessful termination (0). As is done for regular expressions, this operator is sometimes not written.
- Constant 0 denotes *inaction* (or deadlock), and is the neutral element of alternative composition. This constant is denoted δ in ACP-style process algebra [4]. Process 0 cannot execute any action, and cannot terminate. Notation 0 is also used in language theory.
- 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 [4]. Process 1 cannot execute any action, but terminates successfully. 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 [5], 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 BPA$^*_{0,1}$ is given by axioms A1-9 and KS1-3 in Table 1. Axioms A1-9 are standard. Compared to language theory, we do not have the law $x\cdot(y+z) = x \cdot y + x \cdot z$ (the ‘wrong’ distributivity, these terms differ in the moment of choice), and 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). 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 behaviour (in language theory, we say that we can assume that the iterated term does not have the empty word property); taking $x = 0$ yields $1^* = 1$. KS3 is the axiom of Troeger ([19]).

Table 1. Axioms of 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

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

Definition 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 $a \cdot t$, $t^* \cdot s$ and $t + s$ are normal forms.

Proposition 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 We can turn the axioms A3-9 of BPA$^*_{0,1}$ into rewrite rules, by orienting them from left to right. We obtain a confluent and terminating term rewrite system modulo A1-2. Then, reduce t to normal form. The result may still contain summands of the form a (only an atomic action) or a^* (only an iteration). These have to be replaced by $a \cdot 1$ and $a^* \cdot 1$, respectively. This proof is like several examples in [4] or [3]. $\square$

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

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

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. The 1 summand may or may not occur.

We will have need to strengthen this result a little bit. For this, we use a result of Milner, proposition 6.2 of [15]:

Proposition 3 [15]: For each closed BPA$^*_{0,1}$-term t there is an effective algorithm producing a closed BPA$^*_{0,1}$-term s with

$$\text{BPA}^*_{0,1} \vdash t = s$$

and s has no subterm of the form f^* with $f = f + 1$.

Using this result, we can require in the normal form in addition that terms u_j do not satisfy $u_j = u_j + 1$, i.e. these terms do not have the empty word property. This will ensure that the recursive specifications we define further down are guarded.

Next, we provide a model for BPA$^*_{0,1}$ on the basis of structured operational rules (so-called *SOS rules*) in the style of Plotkin (see [17]). The rules in Table 2 define the following relations on closed BPA$^*_{0,1}$-terms: binary relations $\cdot \xrightarrow{a} \cdot$ (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 $\cdot \xrightarrow{a} \surd$ as in [4] or [3].

The rules provide a transition system for each closed term. Next, we define an equivalence relation on the resulting transition systems in the standard way.

Definition 4 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 $R(s, t)$.

Table 2. Transition rules for BPA$^*_{0,1}$ ($a \in A$).

$$1 \downarrow \qquad x^* \downarrow \qquad a \xrightarrow{a} 1$$

$$\frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$$

$$\frac{x \downarrow}{x + y \downarrow} \qquad \frac{y \downarrow}{x + y \downarrow}$$

$$\frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y} \qquad \frac{x \downarrow, y \xrightarrow{a} y'}{x \cdot y \xrightarrow{a} y'} \qquad \frac{x \downarrow, y \downarrow}{x \cdot y \downarrow}$$

$$\frac{x \xrightarrow{a} x'}{x^* \xrightarrow{a} x' \cdot x^*}$$

Proposition 5 Bisimulation equivalence is a congruence relation on closed BPA$^*_{0,1}$-terms.

Proof This is a standard result following from the format of the deduction rules, see e.g. [3]. $\square$

Theorem 6 *The theory BPA$^*_{0,1}$ is sound for the model of transition systems modulo bisimulation, i.e. for all closed terms t, s we have*

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

Proof This is also a standard result. $\square$

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 ([18]). This impossibility is due to the presence of the 0 constant. In the absence of 0, more positive results can be found in [8] and [9] where a complete axiomatization of regular expressions up to bisimulation equivalence is given when the language satisfies the so-called hereditary non-empty word property (essentially requiring that the non-empty word property be satisfied at any depth within a star context).

The first axiom of iteration is a specific instance of a *recursive equation*. It is standard to use recursive equations to specify processes with possible infinite behaviour, see e.g. [4] or [3]. We proceed to define recursion in our setting.

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, BPA$^*_{0,1}$) and variables from V. A *solution* of a recursive specification $E(V)$ in our theory is a set 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.

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 BPA$^*_{0,1}$ will satisfy none, one or both of these principles. Let us look at the transition system models in particular.

Consider a recursive specification E. For each variable X of E, we can add a new constant $\langle X|E\rangle$ to our syntax. Table 3 provides deduction rules for these constants. They come down to looking upon $\langle X|E\rangle$ as the process $\langle t_X|E\rangle$, which is t_X with, for all $Y \in V$, all occurrences of Y in t_X replaced by $\langle Y|E\rangle$. 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 $\langle t_i|E\rangle$ is defined to be $t_i(\langle X_0|E\rangle, \ldots, \langle X_{n-1}|E\rangle)$.

Table 3. Transition rules for recursion ($a \in A$).

$$\frac{\langle t_X|E\rangle \xrightarrow{a} y}{\langle X|E\rangle \xrightarrow{a} y} \qquad \frac{\langle t_X|E\rangle \downarrow}{\langle X|E\rangle \downarrow}$$

Now if we add such constants $\langle X|E\rangle$ for all specifications E to our syntax, we obtain a model $\mathbb{G}^\infty$ for BPA$^*_{0,1}$ that satisfies RDP and RSP (see e.g. [4]). If we add constants $\langle X|E\rangle$ 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, whereas for guarded recursion we can always get a finitely branching solution. 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;

Note that, due to the presence of the constant 1, a difference between BPA$^*_{0,1}$ and the standard theory BPA is that finite guarded recursion allows the specification of a process with unbounded branching (see [7]).

The third 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 two forms

1. $X = 0 + a_1 \cdot X_1 + \cdots + a_n \cdot X_n$, or

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 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 e.g. [6]. 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):

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

Finally, the term model $\mathbb{P}$ of BPA$^*_{0,1}$ is even smaller than $\mathbb{R}$. In [5] it is shown that there are regular processes that cannot be defined just 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 BPA$^*_{0,1}$ provides a complete axiomatization of the model $\mathbb{P}$.

3 Well-Behaved Specifications

We define a class of recursive specifications over $\text{BPA}^*_{0,1}$ that we will call *well-behaved*. The idea is that the class of well-behaved specifications corresponds exactly to the class of closed $\text{BPA}^*_{0,1}$-terms.

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 7 A recursive specification E over $\text{BPA}^*_{0,1}$ 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:
$$X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \ldots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} +$$
$$+ e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho + c,$$
where $m \geq 0$, $e_{\sigma i}, e_\sigma, e_\rho$ are closed $\text{BPA}^*_{0,1}$-terms, $c \in A \cup \{0, 1\}$, 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
$$X_\rho = 1 \cdot X_{\rho 0} + 1 \cdot X_{\rho 1},$$
i.e. $m = 2$, $e_{\rho 0} = e_{\rho 1} = 1$ 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, i.e. elements of $A \cup \{0, 1\}$.

Definition 8 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
$$X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \ldots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} +$$
$$+ e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho,$$
so there is no constant term present, the last two summands are optional, and all $X_{\sigma i}$ cycle back to X_ρ.

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
$$X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \ldots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + c,$$
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 9 *Every well-behaved recursive specification E has a closed term in $\text{BPA}^*_{0,1}$ 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 (for, when we have a closed term for each variable of the specification, these closed terms together make up a solution in $\mathbb{G}$, and since bisimilarity of closed terms on $\mathbb{G}$ and $\mathbb{P}$ coincides, also a solution in $\mathbb{P}$).

In these lemmas, we will use the following notation. Write $X_\sigma \downarrow X_\rho$ if the equation of X_σ contains a summand $e_\rho \cdot X_\rho$ with $e_\rho \downarrow$. $\Downarrow$ denotes the transitive closure of the relation $\downarrow$ on recursion variables. A useful characerization 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 $X_\sigma \not\Downarrow X_\sigma$.

Lemma 10 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 $\text{BPA}^*_{0,1}$ such that $X_\sigma = e \cdot X_\rho$ (here, $=$ denotes derivability in $\text{BPA}^*_{0,1}$ + RDP$^-$ + RSP*). Moreover, if $X_\sigma \not\Downarrow X_\rho$, 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 $X_\sigma = e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho$ or $X_\sigma = e_\sigma \cdot X_\sigma$. By guardedness, $e_\sigma \not\downarrow$. In the first case, we are done immediately, in the second case, it follows from RSP* that $X_\sigma = e_\sigma^* \cdot e_\rho \cdot X_\rho$ and in the third case, we have $X_\sigma = e_\sigma^* \cdot 0 \cdot X_\rho$. When $X_\sigma \not\Downarrow X_\rho$, we must have $e_\rho \not\downarrow$, which implies $e_\sigma^* \cdot e_\rho \not\downarrow$. Always $e_\sigma^* \cdot 0 \not\downarrow$.

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 $X_\sigma \downarrow X_{\sigma 0}$, we must have $X_{\sigma 0} \not\Downarrow X_\sigma$. 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 $X_\sigma \not\Downarrow X_\rho$, then $X_{\sigma 1} \not\Downarrow X_\rho$, 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 $X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \ldots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho$ 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 $X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \ldots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho = e_{\sigma 0} \cdot f_0 \cdot X_\rho + \ldots + e_{\sigma(m-1)} \cdot f_{m-1} \cdot X_\rho + e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho = e_\sigma \cdot X_\sigma + (e_{\sigma 0} \cdot f_0 + \ldots e_{\sigma(m-1)} \cdot f_{m-1} + e_\rho) \cdot X_\rho = e_\sigma^* \cdot (e_{\sigma 0} \cdot f_0 + \ldots e_{\sigma(m-1)} \cdot f_{m-1} + e_\rho) \cdot X_\rho$. When $X_\sigma \not\Downarrow X_\rho$, then $e_\rho \not\downarrow$, and moreover for each $i < m$ we have either $e_{\sigma i} \not\downarrow$ or $X_{\sigma i} \not\Downarrow X_\rho$. 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 $e_{\sigma 0} \cdot f_0 + \ldots e_{\sigma(m-1)} \cdot f_{m-1} + e_\rho \not\downarrow$ and so $e_\sigma^* \cdot (e_{\sigma 0} \cdot f_0 + \ldots e_{\sigma(m-1)} \cdot f_{m-1} + e_\rho) \not\downarrow$.

□

Lemma 11 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 $X_\sigma = e$ (= is again derivability in BPA$^*_{0,1}$ + RDP$^-$ + RSP*).

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 the definition of cycling back there is a closed term f such that $X_{\sigma 0} = f \cdot X_\sigma$. As $X_\sigma \downarrow X_{\sigma 0}$, we must have $X_{\sigma 0} \not\Downarrow X_\sigma$, 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 $X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \ldots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + c$ 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 $X_\sigma = e_{\sigma 0} \cdot X_{\sigma 0} + \ldots + e_{\sigma(m-1)} \cdot X_{\sigma(m-1)} + e_\sigma \cdot X_\sigma + c = e_{\sigma 0} \cdot f_0 + \ldots + e_{\sigma(m-1)} \cdot f_{m-1} + e_\sigma \cdot X_\sigma + c = e_\sigma^* \cdot (e_{\sigma 0} \cdot f_0 + \ldots + e_{\sigma(m-1)} \cdot f_{m-1} + c)$.

□

Note 12 Our notion of cycling was inspired by (but is different from) the notion of ruling in [10]; 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 [1]. In [1], 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 BPA$^*_{0,1}$ (crucial point: $0^* = 1$), 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 BPA$^*_{0,1}$ can be written as $0, 1$ or in the form

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

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 [10]. Take $e = a(a^*b + c) + (c^* + a^*b)^*c^* + a$. This term is in normal form, but the star term $c^* + a^*b$ has the empty word property, as $c^* \downarrow$. Therefore, we rewrite this term to $e' = a(a^*b + c) + (cc^* + a^*b)^*c^* + a$. Actually, the algorithm in Section 2 also rewrites constants a to $a \cdot 1$ and terms a^* to $a^* \cdot 1$, but we ignore this in the example. Associate X_ϵ to e'.

1. $X_\epsilon = aX_0 + 1X_1 + a$. Thus, X_0 is associated to $a^*b + c$, X_1 to $(cc^* + a^*b)^*c^*$.

2. $X_0 = 1X_{00} + c$. Thus, X_{00} is associated to a^*b.

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 $a \cdot X_{00}$, and X_{001} corresponds to b.

4. $X_{000} = aX_{00}$. Variable X_{000} cycles back to X_{00}.

5. $X_{001} = b$.

6. $X_1 = X_{10} + X_{11}$. Again, a star-term is split into two parts. Here, X_{10} corresponds to $(cc^* + a^*b) \cdot X_1$, and X_{11} corresponds to c^*.

7. $X_{10} = cX_{100} + X_{101}$. Here, X_{100} corresponds to $c^* \cdot X_1$, and X_{101} corresponds to $a^*b \cdot X_1$.

8. $X_{100} = X_{1000} + X_{1001}$. Again, a star term.

9. $X_{1000} = cX_{100}$. Variable X_{1000} cycles back to X_{100}.

10. $X_{1001} = X_1$. Variable X_{1001} cycles back to X_1.

11. $X_{101} = X_{1010} + X_{1011}$. Split of star.

12. $X_{1010} = aX_{101}$. Variable X_{1010} cycles back to X_{101}.

13. $X_{1011} = bX_1$. Variable X_{1011} cycles back to X_1.

14. $X_{11} = X_{110} + X_{111}$. Split of star.

15. $X_{110} = cX_{11}$.

16. $X_{111} = 1$.

Note that the resulting recursive specification is guarded. Note that the resulting specification is much more restricted than the general format of well-behaved specifications, in the following way: an equation is required to be in the form

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

$$+ e_\sigma \cdot X_\sigma + e_\rho \cdot X_\rho + c.$$

Here, we have that all expressions $e_{\sigma i}, e_\rho$ are constants, and that term $e_\sigma \cdot X_\sigma$ does not occur (stated differently, we can always take $e_\sigma = 0$). Moreover, we can require that the constant c is either 0 or 1. If X_σ has a summand of the form $a \cdot X_\tau$, we write $X_\sigma \xrightarrow{a} X_\tau$. The notation $X_\sigma \downarrow X_\tau$ used in the previous section now means that X_σ has a summand of the form $1 \cdot X_\tau$. Note that if $X_\sigma \downarrow X_\tau$, then either X_σ or X_τ is a cycling variable. Let us call a well-behaved specification in this restricted form a well-behaved specification *in restricted form*.

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

Proposition 13 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 $e = a_0 \cdot t_0 + \ldots a_{n-1} \cdot t_{n-1} + u_0^* \cdot v_0 + \ldots + u_{m-1}^* \cdot v_{m-1} + \{1\}$ 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. $X_\epsilon = a_0 \cdot X_0 + \ldots a_{n-1} \cdot X_{n-1} + X_n + \cdots + X_{n+m-1} + \{1\}$.

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 $X_{(n+j)1\sigma} \not\Downarrow X_{(n+j)1\sigma}$ as $X_\sigma \not\Downarrow X_\sigma$ in G_j; on the other hand, if for some variable $X_{(n+j)0\sigma}$ in F'_j we would have $X_{(n+j)0\sigma} \Downarrow X_{(n+j)0\sigma}$, this cannot be due to a cycle of 1-steps in F_j by the same argument, so we must have $X_{(n+j)0\sigma} \Downarrow X_{n+j} \Downarrow X_{(n+j)0\sigma}$. This implies $X_{(n+j)0} \Downarrow X_{n+j}$, 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 $X_\epsilon = a_0 \cdot X_0 + \ldots a_{n-1} \cdot X_{n-1} + X_n + \cdots + X_{n+m-1} + \{1\} = a_0 \cdot t_0 + \ldots a_{n-1} \cdot t_{n-1} + u_0^* \cdot v_0 + \cdots + u_{m-1}^* \cdot v_{m-1} + \{1\} = e$. □

Thus, for each closed 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 automaton has a well-behaved recursive specification or not. Suppose we have given a finite transition system. We can assume this system is minimized, i.e. the largest autobisimulation has been divided out. Then, we can assume that all states are not bisimilar. However, this is not necessary for the following procedure.

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 14 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 $X_\sigma \downarrow X_\rho$, 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 $X_\tau \xrightarrow{a} X_{\tau i}$ or $X_\tau \downarrow X_{\tau i}$. 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, consider 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 $X_\tau \xrightarrow{a} X_{\tau i}$, 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 $X_{\sigma\rho}$, i.e. we throw out all equations of variables of the form $X_{\sigma\pi}$, and we put in new equations

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

whenever there was an equation

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

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. But X_σ must be bisimilar to a state or a substate s of the given transition system. This (sub)state has a number of transitions $s \xrightarrow{a} s'$, 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 15 Cycling variables can be nested only $n \cdot 2^{2k}$ deep, i.e. 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 $X_\sigma, X_{\sigma\rho}$ such that $X_{\sigma\rho}$ is in the cycling part of X_σ, i.e. it is below $X_{\sigma 0}$. Now $X_\sigma, X_{\sigma\rho}$ 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 $X_\sigma, X_{\sigma\rho}$ such that $X_{\sigma\rho}$ is below $X_{\sigma 0}$, and $X_\sigma, X_{\sigma\rho}$ 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 $X_{\sigma\rho 0}$, i.e. 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 occur that there is a sequence of cycling variables, that alternatingly occur in the cycling part and in the exit part. I.e. there can be cycling variables $X_\sigma, X_{\sigma\rho}, X_{\sigma\rho\pi}$ 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.

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

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. $\square$

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:

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

6 Star Height

In this section we consider some consequences of the results obtained. In particular, we look at the star height problem. The star height of a regular expression is the maximum number of nested stars it contains. In formal language theory, the notion of star height originates in [11].

Definition 16 Let t be a closed 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. $sh(t^*) = 1 + sh(t)$.

Let t be a closed BPA$^*_{0,1}$-term. We say t has *minimal star height* n iff n is the minimal number such that there is regular expression equal to t (i.e., bisimilar to t) of star height n.

An obvious question is now how to determine the minimal star height of a given regular expression. In language theory, this problem was solved in [12]. In our setting, using bisimulation equivalence instead of language equivalence, we can achieve the same result, by use of well-behaved specifications.

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

Definition 17 Let E be a recursive specification over 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

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

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 Lemma's 10 and 11 can be strengthened as follows, by following the proofs step by step, using the definitions just given.

Lemma 18 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 19 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 $X_\sigma = e$.

Thus, the procedure of Theorem 9 assigns to each well-behaved recursive specification a regular expression of the same star height; 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 20 Let e be a closed term over BPA$^*_{0,1}$. Then there is a well-behaved specification E in restricted form with e as a solution that satisfies $sh(E) = sh(e)$.

With the help of these propositions, the star height problem can be solved.

Theorem 21 *Let e be a closed term over BPA$^*_{0,1}$. There is an algorithm to find the minimal star height of e.*

Proof Let e be given. Let E be the restricted well-behaved specification with solution e given by the procedure in Section 4. By the reductions in Section 5, E can be reduced to satisfy certain bounds, which depend on e, on the branching degree and the length of descending paths in E. Note that none of these reductions increase star height. They can reduce star height, though. The star height of the reduced specification is the star height of some closed term that is bisimilar to e.

Using this observation, it is a consequence of the previous lemma that there exists a well-behaved specification in restricted form below the mentioned bounds that has e as a solution and that has the minimal star height of e as its star height. Therefore it is enough to search through all restricted well-behaved specifications below these bounds for ones that have solution e, and to choose, among the finitely many specifications with these properties, a specification with the least possible star height. It follows from the previous lemmas that the minimal star height of e is equal to the star height of this specification, and from this specification, a closed term can be constructed that is bisimilar to e and that has star height equal to the minimal star height of e. $\square$

Star height gives a hierarchy on regular expressions. In language theory, for any n, there is a regular expression with minimal star height n, as long as the alphabet has at least two letters ($|A| \geq 2$), but the hierarchy collapses if there is just one action (if $|A| = 1$, then the minimal star height of any regular expression is at most 1) (see [11, 14]).

Milner [15] speculates that in the case of bisimulation, the hierarchy is non-trivial, even in the case of a singleton alphabet, and gives a set of regular expressions p_n that should have minimal star height n. Hirshfeld and Moller prove in [13] that this is indeed the case. We give an alternative proof of this fact using our theory.

Proposition 22 Define the set of regular expressions p_n inductively:

- $p_1 = a^*$
- $p_{n+1} = (p_n \cdot a)^*$.

Then the minimal star height of p_n is n.

Proof We give a proof sketch using well-behaved specifications. Consider the following recursive specification:

$$\begin{array}{rcl} Y_n & = & a \cdot Y_1 + \ldots + a \cdot Y_n + 1 \\ Y_{n-1} & = & a \cdot Y_1 + \ldots + a \cdot Y_n \\ & & \ldots \\ Y_1 & = & a \cdot Y_1 + a \cdot Y_2. \end{array}$$

Note that the different variables Y_i are not bisimilar, i.e. the system is minimized. Now the processes p_n, $p_{n-1} \cdot a \cdot p_n$, …, $p_1 \cdot a \cdot \ldots \cdot a \cdot p_n$ form a solution for the variables $Y_n, Y_{n-1}, \ldots, Y_1$. This in turn, can be seen by noticing that in BPA$^*_{0,1}$ plus RSP*, it is derivable that

$$p_n = ap_1 \ldots ap_n + ap_2 \ldots ap_n + \ldots + ap_{n-1}ap_n + ap_n + 1.$$

Now it is easy to see that p_n has star height n. For showing that p_n also has minimal star height n, it suffices to demonstrate that all restricted well-behaved specifications with p_n as solution have star height at least n. Thus, let E be such a specification. Then variable X_ϵ in E is equivalent to variable Y_n in the specification above. It follows that each variable in E is bisimilar to a (sub)state of one of the variables Y_i. It is not difficult to prove that if, for a variable X_σ in E, i is minimal with the property that X_σ is bisimilar to Y_i or to a substate of Y_i, then the star height of X_σ must be at least i. From this, $sh(E) = sh(X_\epsilon) \geq n$ follows. $\square$

Finally notice that if the set of atomic actions is empty, then the set of closed terms reduces to just $\{0, 1\}$, so all closed terms have minimal star height 0.

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 that are expressible by a regular expression (modulo bisimulation). This means we have solved the open question of Milner ([15]).

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. Note that Bosscher 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 $0, 1$ (see [7]).

Our results can be adapted to the setting of [10], where the constant 0 really acts as the zero process.

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 non-trivial, even in the case of a singleton alphabet.

References

[1] J.C.M. Baeten and J.A. Bergstra. Process algebra with a zero object. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR'90*, number 458 in Lecture Notes in Computer Science, pages 83–98. Springer Verlag, 1990.

[2] J.C.M. Baeten and F. Corradini. Regular expressions in process algebra. In *Proceedings LICS'05*, pages 12–19. IEEE Computer Society, 2005. Also report CS-R 05-11, Computer Science, Technische Universiteit Eindhoven.

[3] J.C.M. Baeten and C. Verhoef. Concrete process algebra. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4, pages 149–269. Oxford University Press, 1995.

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

[5] J.A. Bergstra, I. Bethke, and A. Ponse. Process algebra with iteration and nesting. *The Computer Journal*, 37(4):243–258, 1994.

[6] J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In J. Paredaens, editor, *Proceedings 11th ICALP*, number 172 in LNCS, pages 82–95. Springer Verlag, 1984.

[7] D.J.B. Bosscher. *Grammars Modulo Bisimulation*. PhD thesis, University of Amsterdam, 1997.

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

[9] F. Corradini, R. De Nicola, and A. Labella. An equational axiomatization of bisimulation over regular expressions. *Journal of Logic and Computation*, 12:301–320, 2002.

[10] R. De Nicola and A. Labella. Nondeterministic regular expressions as solutions of equational systems. *Theoretical Computer Science*, 203:179–189, 2003.

[11] L.C. Eggan. Transition graphs and the star-height of regular events. *Michigan Mathematics Journal*, 10:385–397, 1963.

[12] K. Hashiguchi. Representation theorems on regular languages. *Journal of Computer System Sciences*, 27:101–105, 1983.

[13] Y. Hirshfeld and F. Moller. On the star height of unary regular behaviours. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language, and Interaction (Essays in Honour of Robin Milner)*, Foundations of Computing, chapter 16, pages 497–509. MIT Press, 2000.

[14] R. McNaughton. The loop complexity of regular events. Technical Report Machine Structures Group Memo 18, MIT, 1988.

[15] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Comput. System Sci.*, 28(3):439–466, 1984.

[16] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[17] G.D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60:17–139, 2004. Reprint from 1981 in Special Issue on Structural Operational Semantics.

[18] P. Sewell. Nonaxiomatisability of equivalences over finite state processes. *Annals of Pure and Applied Logic*, 90:163–191, 1997.

[19] D.R. Troeger. Step bisimulation is pomset equivalence on a parallel language without explicit internal choice. *Math. Struct. Comput. Sci.*, 3(1):25–62, 1993.