Towards a Uniform Theory of Effectful State Machines

SERGEY GONCHAROV, Friedrich-Alexander-Universität Erlangen-Nürnberg
STEFAN MILIUS, Friedrich-Alexander-Universität Erlangen-Nürnberg
ALEXANDRA SILVA, University College London

Using recent developments on coalgebraic and monad-based semantics, we present a uniform study of various notions of machines, e.g. finite state machines, multi-stack machines, Turing machines, valence automata, and weighted automata. They are instances of Jacobs' notion of a $\mathbf{T}$-*automaton*, where $\mathbf{T}$ is a monad. We show that the generic language semantics for $\mathbf{T}$-automata correctly instantiates the usual language semantics for a number of known classes of machines/languages, including regular, context-free, recursively-enumerable and various subclasses of context free languages (e.g. deterministic and real-time ones). Moreover, our approach provides new generic techniques for studying the expressivity power of various machine-based models.

CCS Concepts: •**Theory of computation** → **Grammars and context-free languages; Quantitative automata; Regular languages; Categorical semantics;**

Additional Key Words and Phrases: monads, side-effects, coalgebras, bialgebraic semantics, Kleene theorem

ACM Reference Format:
Sergey Goncharov, Stefan Milius and Alexandra Silva, 2016. Towards a Uniform Theory of Effectful State Machines. *ACM Trans. Comput. Logic* 00, 00, Article 00 (2018), 61 pages.
DOI: 0000001.0000001

1. INTRODUCTION

In recent decades much interest has been drawn to studying generic abstraction devices that not only formally generalize various computation models and tools, but also help to identify core principles and reasoning patterns behind them. One example of this kind is given by the notion of *computational monad* [Moggi 1991], which made an impact both on the theory of programming (as an organization tool for denotational semantics [Fiore et al. 2002; Plotkin and Power 2002]) and on the practice (e.g. being implemented as a programming language feature of Haskell [Peyton Jones 2003] and F# [Syme et al. 2007]). Another pivotal abstraction device is given by the notion of *coalgebra*, providing a uniform syntax-independent framework for concurrency theory and observational semantics of state based systems (see e.g. [Rutten 2000]).

In this paper, we combine the use of monads and coalgebras for formalizing semantics and behaviors of systems to give a unified (bialgebraic) perspective of classical automata theory as well as of some less standard models such as weighted automata and valence automata.

We base our framework on the notion of $\mathbf{T}$-*automaton* whose original definition goes back to [Jacobs 2006]. A $\mathbf{T}$-automaton is a coalgebra of the form

$$m : X \to B \times (TX)^A,$$

where T is the functor part of a monad $\mathbf{T}$, which we understand as a mathematical abstraction of a *computational effect* (in the sense of [Moggi 1991]) happening in conjunction with state transitions

Stefan Milius acknowledges support by Deutsche Forschungsgemeinschaft (DFG) under project MI 717/5-1.
Sergey Goncharov acknowledges support by Deutsche Forschungsgemeinschaft (DFG) under project GO 2161/1-2.
Author's addresses: S. Goncharov., S. Milius, Friedrich-Alexander-Universität Erlangen-Nürnberg, Chair for Theoretical Computer Science, Martenstraße 3, 91058 Erlangen, Germany; Alexandra Silva, Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom.

DOI: 0000001.0000001

of the automaton, A is the set of inputs, and B is the set of outputs which is required to be a T-algebra. For example, nondeterminism, viz. the computational effect of nondeterministic finite state machines, is modeled by the finite-power set monad $T = \mathcal{P}_\omega$, and $B = \{0, 1\}$, modeling accepting and rejecting states is a $\mathcal{P}_\omega$-algebra, equivalently a join-semilattice, with the obvious structure. Analogously, we show that certain (nondeterministic) extensions of the pushdown store form the underlying effect of pushdown automata.

A crucial ingredient of our framework is the *generalized powerset construction* [Silva et al. 2013], which serves as a coalgebraic counterpart of classical Rabin-Scott determinization algorithm [Rabin and Scott 1959] and allows us to provide a generic *(deterministic) semantics* of T-automata. By instantiating the operational analysis of computational effects from [Plotkin and Power 2002] to our setting we axiomatize relevant monads and algebras and thus arrive at syntactic fixpoint expressions, which we dub *reactive expressions*, representing T-automata. Furthermore, we prove a Kleene-style theorem relating T-automata and the corresponding expressions, thus generalizing previous work in [Silva et al. 2010; Silva et al. 2011]. This generic correspondence instantiates to three large classes of machines actively studied in the literature:

— state machines over various types of store, as classically studied in formal language theory [Rozenberg and Salomaa 1997]; here we elaborate in detail push-down stores and their combination with one another and with nondeterminism, as well as Turing tapes;
— valence automata [Render and Kambites 2009; Kambites 2009; Zetzsche 2016], capturing nondeterministic computations over a store modeled by various classes of monoids;
— weighted automata [Droste et al. 2009; Sakarovitch 2009].

We also capture systems combining probability and nondeterminism [Segala 1995; Segala and Lynch 1995], which do not fit any of the above classes.

A unifying semantic domain in our framework is the set B^{A^*} of *formal power series*, standardly used in weighted automata theory (where B is assumed to be a semiring). With B being the two-element set $\{0, 1\}$ this is isomorphic to the set of all formal languages over A, which is the semantic domain for finite state automata. In the case of *stack* T*-automata*, i.e. where T models a pushdown store, B consists of certain predicates in 2^{Γ^*}, where Γ denotes the stack alphabet. Hence formal power series may be identified with certain functions $\Gamma^* \to 2^{A^*}$, and our semantics assigns to a state of a given T-automaton the function which maps a word $w \in \Gamma^*$ to the language recognized by the automaton with initial stack content w. Analogous considerations apply to T-automata where T models a Turing tape. Furthermore, note that most textbooks (e.g. [Hopcroft et al. 2006]) define a Turing machine with a single tape both for performing computations and for communicating the data. However, it is important in our approach to diffentiate the *reactive* and *computational* parts of a machine. Therefore we consider *online Turing machines* [Hennie 1966] that have a designated (one-way) input tape alongside with the Turing tape. Essentially the same type of machines (but subject to *bisimulation semantics* instead of *language semantics*) was recently studied under the name *reactive Turing machines* [Baeten et al. 2011].

The format of our general reactive expressions deviates from the format of the familiar Kleene's regular expressions. This is inevitable, for the latter use various features of the underlying model that are not generally available, most notably nondeterministic choice, but also the fact that B is precisely the two-element set $\{0, 1\}$. However, our syntax features precisely the operations coming from an equational presentation of the computation monad T. This allows us to instances which are beyond the reach of expression formats with "hard-wired" nondeterminism. Specifically, we elaborate the case of deterministic machines over a pushdown store, recognizing precisely *real-time deterministic context-free languages*, which are properly contained in the class of all context-free languages, which in turn are recognized by the respective nondeterministic stack T-automata. Moreover, we show that our syntax can be simplified for monads whose presentation features a finitary summation operation (generalizing nondeterministic choice), and under further expected assump-

tions, become convertible to the one familiar in weighted automata theory for defining *rational* formal power series [Droste et al. 2009; Sakarovitch 2009].

A considerable part of our technical development (especially Section 3) is devoted to characterizing monads capturing realizable transitions of state machines. For example, the stack of a pushdown automaton is standardly modeled by the set of finite sequences Γ^* over an alphabet Γ of stack symbols. However, not every transformation $\Gamma^* \to \Gamma^*$ is realizable by such an automaton (they need not even be computable). We characterize the relevant *stack monad* of realizable stack transformers in two complementary ways: as a submonad of the store monad $TX = (X \times \Gamma^*)^{\Gamma^*}$ and as an algebraic theory over primitive stack operations $push$ and pop. Analogous characterizations are established for the *(Turing) tape monad*, whose theory, in contrast to the stack theory, fails to be finitely axiomatizable.

The main salient feature of our approach is that it allows one to untie from the standard enumerative and diverse definitions of various kinds of state machines and reason about them collectively in a uniform way. We demonstrate this by providing some initial constructions on $\mathbf{T}$-automata, specifically by *tensoring* the underlying monads for obtaining machines over combined effects, e.g. store and nondeterminism. Another construction we present is a certain *continuations passing style (CPS) transformation* of a given $\mathbf{T}$-automaton allowing us to define an extension of the canonical coalgebraic semantics to the case of unobservable (aka *silent*) transitions. The latter semantics allows us to capture recursively enumerable languages by (deterministic) $\mathbf{T}$-automata over the Turing tape. This provides an answer to a long standing challenge of giving a coalgebraic description for any Turing complete computation model.

Using a reduction to previous work [Book and Greibach 1970] on *real-time machines* we show that $\mathbf{T}$-automata with nondeterminism and an arbitrary number of stacks but *without* unobservable moves capture precisely the class $\mathsf{NTIME}(n)$ of nondeterministic linear time languages. Based on this result we argue that it seems unlikely to be able to capture languages beyond $\mathsf{NTIME}(n)$ by any computationally feasible class of $\mathbf{T}$-automata without unobservable moves. In fact, we conjecture that this bound remains valid also for our tape $\mathbf{T}$-automata. The requirement to be real-time is an inherent feature of coalgebraic modelling and is often regarded a desirable feature of *reactivity* or *productivity* of computations.

Finally, we prove a coalgebraic version of one direction of the classical *Chomsky-Schützenberger theorem* (Theorem 7.5). As an instance, this allows to conclude that for every polycyclic monoid M of rank at least 2, every context-free language is recognized by a valence automaton over M; that context-free languages are precisely the languages recognized by valence automata over polycyclic monoids was proven in [Render and Kambites 2009].

Related work. We build on previous work on coalgebraic modelling and monad-based semantics. Most of the applications of coalgebra to automata and formal languages however address rational models (e.g. rational streams, regular languages) from which we note [Rutten 2003] (regular languages and finite automata), [Jacobs 2006] (bialgebraic treatment of Kleene algebra and regular expressions), [Silva et al. 2010; Silva et al. 2011; Milius 2010; Bonsangue et al. 2013] (coalgebraic regular expressions).

More recently, some further generalizations were proposed. In recent work [Winter et al. 2013] a coalgebraic model of context-free grammars is given, and [Bonsangue et al. 2012] captures weighted context-free grammars and algebraic formal power-series coalgebraically, without however an analogous treatment of (weighted) push-down automata. Winter [2014] devotes a chapter of his thesis to the treatmeant of push-down automata (and weighted push-down systems), including e.g. a bisimulation-based proof of the result that any power series recognizable by a weighted pushdown system is also recognizable by a weighted pushdown system with a single state, the latter of which coincide with weighted grammars in Greibach normal form. However, a final coalgebra based semantics of push-down systems, like the one we present for stack $\mathbf{T}$-automata, is not presented in loc. cit. Finally, [Milius et al. 2016] gives a unifying account of various finite state behaviours, and in particular characterizes the domain of finite state behaviours by a universal property; applications

include all known coalgebraic models of rational behaviour, but also (weighted) context-free languages and algebraic power-series and the languages recognized by T-automata. Myers established a rather general form of a Kleene theorem for surjection preserving functors on varieties [Myers 2013], while we stick to a concrete functor $B \times (-)^A$. His Kleene Theorem is parametric in a given presentation of the variety and the type functor by operations and equations; but we do not derive our Kleene-type theorem from his general one. The specific form of the functor we are using allows us to directly associate T-automata and the corresponding expressions with their semantics, which are formal power series from B^{A^*}. Moreover, this enables us to give a direct syntactic translation between the *reactive expressions* in Section 4 and the more convenient *additive expressions* in Section 5 (see Proposition 5.5).

The notion of T-automata appeared for the first time in [Jacobs 2006]. In addition, we will also use in our development two results from [Jacobs 2006] (these appeared also in Bartels' thesis [Bartels 2004] and Turi and Plotkin's seminal paper [Turi and Plotkin 1997]) stating that: (i) in the presence of a distributive law $\mathsf{T}G \Rightarrow G\mathsf{T}$, the final G-coalgebra carries a T-algebra structure; (ii) there is a bijective correspondence between GT-coalgebras (in $\mathbf{Set}$) and λ-bialgebras. [Jacobs 2006] gives a list of T-automata examples, including non-deterministic automata and semiring automata, but these are not treated in detail and, more importantly, this list does not include machines with memory such as pushdown automata. We go beyond [Jacobs 2006] both in terms of examples, but more importantly, in that we provided a uniform expression syntax for a large class of automata, which include automata equipped with memory, for which we make use of algebraic presentations of monads.

Pattinson and Schöder [2016] independently investigated an axiomatization of the Turing tape equivalent to ours and showed that the axioms precisely characterize the Turing tape as a *final comodel* of the corresponding algebraic theory. They proved a completeness theorem which can be read as the fact that the induced monad injectively embeds into the store monad with the Turing tape as the store. In contrast to the latter result in our work we additionally characterize precisely that submonad by a collection of conditions on the store transformers.

The present paper is a considerably extended version of our previous conference publication [Goncharov et al. 2014].

2. DETERMINISTIC MOORE AUTOMATA, COALGEBRAICALY

In this section we recall the main definitions and existing results on coalgebraic modelling of state machines that we need. This material, as well as the material of the following sections, uses the language of category theory, hence we assume readers to be familiar with basic notions. We use $\mathbf{Set}$ as the main underlying category throughout. Further abstraction from $\mathbf{Set}$ to a more general category, while possible (and often quite straightforward), will not be pursued in this paper.

Our central notion is that of an *F-coalgebra*, where F is an endofunctor on $\mathbf{Set}$ called *transition type*. An F-coalgebra is a pair $(X, f : X \to FX)$ where X is a set called the *state space* and f is a map called *transition structure*. We shall often identify a coalgebra with its state space if no confusion arises.

Coalgebras of a fixed transition type F form a category whose morphisms are maps of the state spaces commuting with the transition structure: a map $h\colon X \to Y$ is a *(coalgebra) homomorphism* from $(X, f\colon X \to FX)$ to $(Y, g\colon Y \to FY)$ if the square below commutes:

$$\begin{array}{ccc} X & \xrightarrow{f} & FX \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle Fh} \\ Y & \xrightarrow[g]{} & FY \end{array}$$

$g \circ h = Fh \circ f$. A final object of this category (if it exists) plays a particularly important role and is called *final coalgebra*. We denote the final F-coalgebra by

$$(\nu F, \iota \colon \nu F \to F\nu F),$$

and write $\widehat{f} \colon X \to \nu F$ for the unique homomorphism from (X, f) to $(\nu F, \iota)$.

Our core example is the standard formalization of Moore automata as coalgebras [Rutten 2000]. For the rest of the paper we fix a finite set A of *actions* and a set B of *outputs*. We call the functor $L = B \times (-)^A$ the *language functor* (over A, B). The coalgebras for L are given by a set X of states with a transition structure on X given by maps

$$o : X \to B \qquad \text{and} \qquad \partial_a : X \to X, \qquad (a \in A)$$

where the left-hand map, called the *observation map*, represents an output function (e.g. an acceptance predicate if $B = 2$; here and elsewhere we identify 2 with $\{0, 1\}$) and the right-hand maps, called *a-derivatives*, are the next state functions indexed by input actions from A. Finite L-coalgebras are hence precisely classical Moore automata. It is straightforward to extend a-derivatives to w-derivatives with $w \in A^*$ by induction: $\partial_\epsilon(x) = x$; $\partial_{aw}(x) = \partial_a(\partial_w(x))$ where $\epsilon \in A^*$ is the empty word.

The final L-coalgebra νL always exists and is carried by the set of all *formal power series* B^{A^*}. The transition structure on B^{A^*} is given by

$$o(\sigma) = \sigma(\epsilon) \qquad \text{and} \qquad \partial_a(\sigma) = \lambda w.\, \sigma(aw), \qquad (a \in A)$$

for every formal power series $\sigma : A^* \to B$. The unique homomorphism from an L-coalgebra X to the final one B^{A^*} assigns to every state $x_0 \in X$ a formal power series that we regard as the *(language) semantics* of X with x_0 as an initial state. Specifically, if $B = 2$ then finite L-coalgebras are deterministic automata and $B^{A^*} \cong \mathcal{P}(A^*)$ is the set of all formal languages over A and the language semantics assigns to every state of a given finite deterministic automaton the language accepted by that state. The transition structure on $\mathcal{P}(A^*)$ is given by the predicate o distinguishing languages containing the empty word and by the maps ∂_a assigning to a language their left derivatives:

$$o(L) = \top \iff \epsilon \in L \qquad \text{and} \qquad \partial_a(L) = \{w \mid aw \in L\} \qquad (a \in A).$$

Definition 2.1 (*Language semantics, Language equivalence*). Given an L-coalgebra (X, f), the *language semantics* is given by

$$\widehat{f} : X \to B^{A^*}$$

For every $x \in X$, $\widehat{f}(x)$ is the formal power series recognized by x.

Language equivalence identifies exactly those x and y for which $\widehat{f}(x) = \widehat{g}(y)$ (for possibly distinct coalgebras (X, f) and (Y, g)); this is denoted by $x \sim y$.

We obtain the following characterization of language equivalence.

PROPOSITION 2.2. *Given $x \in X$ and $y \in Y$ where X and Y are L-coalgebras, $x \sim y$ iff for any $w \in A^*$, $o(\partial_w(x)) = o(\partial_w(y))$.*

PROOF. Let f and g be the transition structures of X and Y, respectively. Since both $\widehat{f}$ and $\widehat{g}$ are L-coalgebra morphisms, we have $o(z) = o(\widehat{f}(z))$ and $\widehat{f}(\partial_a(z)) = \partial_a(\widehat{f}(z))$ for every $a \in A$ and similarly for $\widehat{g}$. By an easy induction, the latter equation yields $\widehat{f}(\partial_w(z)) = \partial_w(\widehat{f}(z))$ for every $z \in X$ and $w \in A^*$. Therefore,

$$\begin{aligned} o(\partial_w(x)) &= o(\widehat{f}(\partial_w(x))) = o(\partial_w(\widehat{f}(x))) = \widehat{f}(x)(w), \\ o(\partial_w(y)) &= o(\widehat{g}(\partial_w(y))) = o(\partial_w(\widehat{g}(y))) = \widehat{g}(y)(w), \end{aligned}$$

where the last equations easily follow from the definitions of o and ∂_w on $\nu L = B^{A^*}$.

Now note that $x \sim y$ iff $\widehat{f}(x) = \widehat{g}(y)$ and the latter holds iff $\widehat{f}(x), \widehat{g}(y) : A^* \to B$ are equal on every $w \in A^*$. Thus we conclude that $o(\partial_w(x)) = o(\partial_w(y))$ iff $x \sim y$ as desired. □

It is well-known that Moore automata, i.e. *finite* L-coalgebras, can be characterized in terms of formal power series occurring as their language semantics (see e.g. [Rutten 2003]).

Definition 2.3 (*Regular power series*). We call a formal power series σ *regular* if the set $\{\partial_w(\sigma) \mid w \in A^*\}$ is finite.

The following result is a rephrasing of a classical result on regular languages (see e.g. [Eilenberg 1974, Theorem III.8.1]). The proof for formal power series is similar and left to the reader.

PROPOSITION 2.4. *A formal power series is accepted by a Moore automaton if and only if it is regular.*

Remark 2.5. Formal power series are usually considered when B is a semiring, in which case one usually also speaks of *recognizable formal power series* as behaviours of finite weighted automata over B (see e.g. [Droste et al. 2009]). Our notion of *regular formal power series* (Definition 2.3) generally disagrees with the latter one (unless B is finite) and is in conceptual agreement with such notions as 'regular events' and 'regular trees' [Goguen et al. 1977; Courcelle 1983].

Regular formal power series as the semantics of precisely the finite L-coalgebras are a special instance of a general coalgebraic phenomenon [Adámek et al. 2006; Milius 2010]. Let F be any finitary endofunctor on $\mathbf{Set}$. Define the set ϱF to be the union of images of all *finite* F-coalgebras $(X, f : X \to FX)$ under their respective unique homomorphisms $\widehat{f} : X \to \nu F$. Then ϱF is a subcoalgebra of νF with an isomorphic transition structure map; ϱF is therefore called the *rational fixpoint* of F. It is (up to isomorphism) uniquely determined by either of the two following universal properties: (1) as an F-coalgebra it is the final locally finite coalgebra and (2) as an F-algebra it is the initial iterative algebra. We refer to [Adámek et al. 2006; Milius 2010] for details.

The characteristic property of regular formal power series can be used as a definitional principle. In fact, given a regular power series σ and assuming that $A = \{a_1, \ldots, a_n\}$, we can view $\{\sigma_1, \ldots, \sigma_k\} = \{\partial_w(\sigma) \mid w \in A^*\}$ as a formal solution of a system of recursive equations of the form

$$\sigma_i = a_1.\sigma_{i_1} \pitchfork \ldots \pitchfork a_n.\sigma_{i_n} \pitchfork c_i, \qquad i = 1, \ldots, k, \tag{2.1}$$

where for all $1 \le i \le k$ and $1 \le j \le n$ we have $\partial_{a_j}(\sigma_i) = \sigma_{i_j}$ and $\sigma_i(\epsilon) = c_i$. Here we introduce $\pitchfork$ as a notation allowing us to syntactically glue together the information about the "head" of a regular formal series and all its derivatives. Reading the $\sigma_1, \ldots, \sigma_k$ as recursion variables, the system (2.1) uniquely determines the corresponding regular power series: for every i it defines $\sigma_i(\epsilon)$ as c_i and for $w = au$ it reduces calculation of $\sigma_i(w)$ to calculation of some $\sigma_j(u)$ – this induction is obviously well-founded.

Any recursive equation system (2.1) can be rewritten as a term using the fixpoint operator μ. To do this, first write

$$\sigma_i = \mu\sigma_i.\, a_1.\sigma_{i_1} \pitchfork \ldots \pitchfork a_n.\sigma_{i_n} \pitchfork c_i \tag{2.2}$$

where $\mu\sigma_i$ binds the occurrences of σ_i in the right-hand term. One can then successively eliminate all the variables σ_i using the equations (2.2) as assignments and thus obtain a syntactic description of the given regular power series as $\sigma = t$ where t is a closed term given by the following grammar:

$$\gamma ::= \mu x.\, a.\delta \pitchfork \ldots \pitchfork a.\delta \pitchfork b \qquad \delta ::= x \mid \gamma \qquad (a \in A, x \in X, b \in B) \tag{2.3}$$

Here X refers to an infinite stock of recursion variables. The term t according to (2.3) is then nothing but a condensed representation of the system (2.1) and as such it uniquely defines σ. Thus every regular formal power series yields a closed term. Proposition 2.6 below together with Poposition 2.4 then establish that closed expressions according to (2.3) capture precisely regular formal power

$$\begin{cases} q_0 = a.q_1 \pitchfork b.q_2 \pitchfork 1 \\ q_1 = a.q_2 \pitchfork b.q_0 \pitchfork 2 \\ q_2 = a.q_0 \pitchfork b.q_1 \pitchfork 3 \end{cases}$$

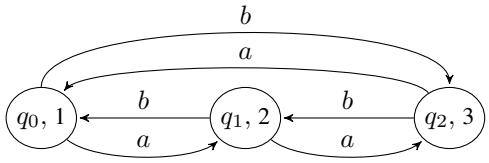


Fig. 1. A Moore automaton over $A = \{a, b\}$, $B = 3 = \{1, 2, 3\}$ as a graph (right) and as the corresponding system of equations (left).

series; this can be viewed as a coalgebraic reformulation of Kleene's theorem. This view has been advanced recently (in a rather more general form) in [Silva et al. 2010; Silva et al. 2011; Myers 2013] and is of crucial importance for the present work.

Admittedly, the expressions of the form (2.3) are still quite close to Moore automata. However, for T-automata (introduced in Section 4) we shall extended this syntax with operations from an algebraic theory given by the monad T (Definition 4.8) and show how to simplify that syntax in the case where T is an additive monad (Definition 5.4); in the special case of weighted automata, this yields a syntax that is equivalent to the familiar rational expressions (Remark 5.6).

Proposition 2.4 in conjunction with the presentation of regular formal power series as expressions (2.3) suggest that every expression gives rise to a finite L-coalgebra generated by it, whose state space consists of expressions. This is indeed true and can be viewed as a coalgebraic counterpart of Brzozowski's classical theorem for regular expressions [Brzozowski 1964]. Given $e = \mu x.\, a_1.e_1 \pitchfork \ldots a_n.e_n \pitchfork c$, let

$$o(e) = c \qquad \text{and} \qquad \partial_{a_i}(e) = e_i[e/x]. \tag{2.4}$$

PROPOSITION 2.6. *Let e be a closed expression (2.3). Then the set $\{\partial_w(e) \mid w \in A^*\}$ forms a finite L-coalgebra under the transition structure defined by (2.4).*

PROOF. We only have to show that $E = \{\partial_w(e) \mid w \in A^*\}$ is finite. Let S be the set of all closed expressions $u\rho$ where u is a subexpression of e and ρ is a substitution sending free variables of u to closed subexpressions of e. Then, S is closed under a-derivatives, for

$$\partial_{a_i}(u\rho) = u_i\rho[u/x] \qquad \text{if } u = \mu x.\, a_1.u_1 \pitchfork \ldots \pitchfork a_n.u_n \pitchfork c,$$

and for $u = x \in X$, we have $\partial_{a_i}(u\rho) = \partial_{a_i}(\rho(x))$, which lies in S by the previous case because $\rho(x)$ is a closed subexpression of e, which must start with a μ-operator. By definition, $e \in S$, hence $E \subseteq S$. Since S is finite, so is E. □

Remark 2.7. If $B = 2$, then Proposition 2.6 is essentially equivalent to Brzozowski's theorem, for in that case the expressions (2.3) are equivalently convertible into the standard regular expressions; the proof of the latter conversion is similar to the one found in [Silva 2010]. The conversion from regular expressions to μ-expressions deploys a determinization procedure, which is available for the underlying notion of automaton. We revisit the question of converting μ-expressions into generalized regular expressions in a broader context in Section 6.

We close this section with a small illustration of the presented material.

Example 2.8. Let $B = \{1, 2, 3\}$ and let $A = \{a, b\}$. Consider a Moore automaton over these data as depicted in Fig. 1. Besides the standard pictorial representation as a graph, we consider an equivalent representation as a system of recursive equations. Given $w \in A^*$ let $\sharp_a w$ and $\sharp_b w$ denote the number of occurrences of a and b in w, respectively. Then the power series σ recognized by state q_i is the one for which

$$\sigma(w) = (\sharp_a w + 2 \cdot \sharp_b w + i) \bmod 3 + 1.$$

After picking q_0 as the initial state we can fold the system of equations into a single fixpoint expression

$$q_0 = \mu x.\, a.\mu y.\, (a.\mu z.\, (a.x \pitchfork b.y \pitchfork 3) \pitchfork b.x \pitchfork 2) \pitchfork \\ b.\mu z.\, (a.x \pitchfork b.\mu y.\, (a.z \pitchfork b.x \pitchfork 2) \pitchfork 3) \pitchfork 1.$$

If we replace 1 in B with $\top$ and both 2 and 3 with $\bot$, then we obtain a deterministic automaton in which q_0 is the only final state. This state then accepts exactly those words $w \in A^*$ for which $\sharp_a w + 2 \cdot \sharp_b w$ is divisible by 3.

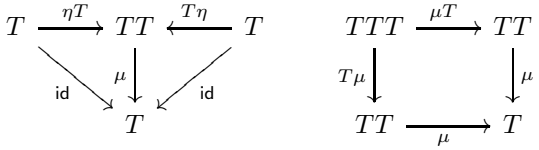
3. MONADS AND Σ-THEORIES

In the previous section we summarized a coalgebraic presentation of deterministic Moore automata, essentially capturing regular languages and regular formal power series. In order to capture bigger language classes we introduce *(finitary) monads* and Σ*-theories* as a critical ingredient of our formalization; this is following and extending ideas in previous work [Jacobs et al. 2012; Silva et al. 2013]. In this work we find it easiest to work with monads in the form of *Kleisli triples*.

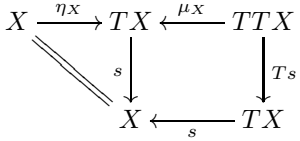
Definition 3.1 (*Kleisli triple*). A Kleisli triple $(T, \eta, \text{–}^\star)$ consists of an object assignment T sending sets to sets, a set-indexed family of maps $\eta_X : X \to TX$ and an operator, called *Kleisli lifting*, sending any map $f : X \to TY$ to $f^\star : TX \to TY$. These data are subject to the following axioms:

$$\eta^\star = \mathsf{id}, \qquad f^\star \cdot \eta = f, \qquad (f^\star \cdot g)^\star = f^\star \cdot g^\star.$$

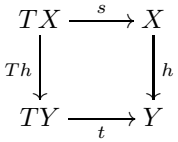
It is well-known that the definition of a monad as a Kleisli triple is equivalent to the usual definition of a monad $\mathbf{T}$ as an endofunctor T equipped with natural transformations $\eta : Id \to T$ (*unit*) and $\mu : TT \to T$ (*multiplication*) making the following diagrams commute:



A $\mathbf{T}$-*algebra* is a pair (X, s) where X is a set (called the *carrier*) and $s : TX \to X$ a map (called the *structure*) such that the diagrams below commute:



and a morphism of $\mathbf{T}$-algebras is just a morphism of algebras for the functor T, i.e. a morphism from (X, s) to (Y, t) is a map $h : X \to Y$ such that the square below commutes:



The category of $\mathbf{T}$-algebras and their morphisms is called *Eilenberg-Moore category of* $\mathbf{T}$ and is denoted by $\mathbf{Set}^{\mathbf{T}}$. Note that (TX, μ_X) is the *free* $\mathbf{T}$*-algebra* on the set X; that means that for every map $f : X \to Y$, where Y is the carrier set of a $\mathbf{T}$-algebra (Y, t), there exists a unique $\mathbf{T}$-algebra morphism $f^\sharp : (TX, \mu_X) \to (Y, t)$ extending f, i.e. such that $f^\sharp \cdot \eta_X = f$. For more background material on monads and $\mathbf{T}$-algebras see [MacLane 1998].

We find it useful to consider monads not only as a technical tool, but also as a metaphor for a notion of computation as manifested by Moggi [1991]. We therefore rely on the syntax of Moggi's *computational metalanguage* (aka, Haskell do-notation):

Notation 3.2 (do-*notation*). For any $p \in TX$ and $q : X \to TY$ we write

$$\mathsf{do}\ x \leftarrow p; q(x)$$

to denote $q^\star(p) \in TY$.

Intuitively, the construction $\mathsf{do}\ x \leftarrow p; q(x)$ should be read as follows: run the computation p; bind the result to x and then run the computation $q(x)$ depending on x. This becomes particularly suggestive when considering state-based monads, for which one can form expressions like

$$\mathsf{do}\ x \leftarrow \mathit{get}(l_1); \mathit{set}(l_2, f(x)),$$

meaning: get a value under location l_1, apply f to it and put the result under l_2.

Remark 3.3. Some comments regarding the do-notation are in order.

(1) The interpretation of the do-notation in general requires that the corresponding monad is *strong*, i.e. equipped with a natural transformation $\tau_{X,Y} : X \times TY \to T(X \times Y)$ called *strength* and satisfying a number of obvious coherence conditions, which are elided here because every monad on $\mathbf{Set}$ is strong via the following canonical strength [Kock 1972]: $\tau_{X,Y}(x, p) = T(\lambda y.\, \langle x, y\rangle)\,(p)$. Strength is needed for propagating values along the do-expressions. For example, the meaning of

$$\mathsf{do}\ x \leftarrow p; y \leftarrow q(x); r(x, y) \quad \text{for every } p \in TX,\ q : X \to TY \text{ and } r : X \times Y \to TZ$$

is precisely $r^\star((\tau_{X,Y}\langle \mathsf{id}_X, q\rangle)^\star(p))$ (which is $\big(\lambda x.\, (\lambda y.\, r(x, y))^\star(q(x))\big)^\star(p)$ in $\mathbf{Set}$).

(2) Further standard notational conventions are as follows:

Notation	Meaning	Condition
$\mathsf{do}\ x \leftarrow p; q$	$\mathsf{do}\ x \leftarrow p; (\lambda x.\, q)(x)$	–
$\mathsf{do}\ p; q$	$\mathsf{do}\ x \leftarrow p; q$	x not a free variable in q;
$\mathsf{do}\ \langle x, y\rangle \leftarrow p; q(x, y)$	$\mathsf{do}\ z \leftarrow p; q(z)$	for $p \in T(X_1 \times X_2)$ and $q : X_1 \times X_2 \to TY$;
$\mathsf{do}\ x_1 \leftarrow p_1; \ldots; x_n \leftarrow p_n; q$	$\mathsf{do}\ x_1 \leftarrow p_1; \ldots; \mathsf{do}\ x_n \leftarrow p_n; q$	–

(3) Moggi [1991] has indeed proved that the following axiomatization of do-expressions is sound complete for strong monads

$$\begin{aligned} \mathsf{do}\ x \leftarrow (\mathsf{do}\ y \leftarrow p; q); r &= \mathsf{do}\ y \leftarrow p; x \leftarrow q; r && (y \text{ not free in } r)\\ \mathsf{do}\ x \leftarrow \eta_X(a); p &= p[a/x]\\ \mathsf{do}\ x \leftarrow p; \eta_X(x) &= p. \end{aligned}$$

making the do-notation a fully fledged *internal language* of strong monads.

A monad T is *finitary* if the underlying functor T is finitary, i.e., T preserves filtered colimits. Informally, T being finitary means that T is determined by its action on finite sets. In addition, finitary monads admit a presentation in terms of (finitary) equational theories over an algebraic signature as we now outline.

Definition 3.4 (Σ-*theory*). An *algebraic signature* Σ consists of operation symbols f, each of which comes together with its *arity* n, which is a nonnegative integer – we denote this by $f : n \to 1$. Symbols of zero arity are also called *constants*. Σ-terms are constructed from the operations in Σ and variables in the usual way. A Σ-*theory* is given by a set of Σ-term equations closed under inference of the standard equational logic. We shall usually present an algebraic theory $\mathcal{E}$ by its signature Σ together with a set of *axioms*; we then obtain $\mathcal{E}$ as the *deductive closure* of the given set of axioms under standard equational reasoning.

Given a Σ-theory $\mathcal{E}$ we can form a monad $\mathbf{T}_\mathcal{E}$ as follows: $T_\mathcal{E}X$ is the set of equivalence classes of terms of the theory over free variables from X (in what follows we shall refer to equivalences of terms always by terms representing them); $\eta_X : X \to T_\mathcal{E}X$ casts a variable to a term; given $\rho : X \to T_\mathcal{E}Y$ and $p \in T_\mathcal{E}X$, $\rho^\star(p)$ is the term $p\rho$ obtained by substituting the free variables in the term p according to the substitution ρ.

Conversely, we can pass from a finitary monad $\mathbf{T}$ to the $\Sigma_\mathbf{T}$-theory $\mathcal{E}_\mathbf{T}$, where $\Sigma_\mathbf{T}$ is the signature that contains an operation symbol $f_a : n \to 1$ for each element a of Tn. Such an operation symbol can be interpreted as a map

$$\langle t_1, \dots, t_n \rangle \mapsto (\lambda i.\, t_i)^\star(a)$$

from $(TX)^n$ to TX. This yields a semantics of $\Sigma_\mathbf{T}$-terms over TX and we define $\mathcal{E}_\mathbf{T}$ to be the $\Sigma_\mathbf{T}$-theory given by all term equations valid over any TX. Notably, $\mathbf{T}$-algebras are then exactly the models of the Σ-theory $\mathcal{E}_\mathbf{T}$.

While the passage from a monad to the $\Sigma_\mathbf{T}$-theory $\mathcal{E}_\mathbf{T}$, followed by the passage in the opposite direction yields an identical transformation, the passage from a Σ-theory, followed by the passage from monads to theories does not yield the original Σ-theory, but instead produces its *clone*, i.e. a theory, obtained from the original Σ-theory by recognizing all Σ-terms as (possibly new) operation symbols. This fundamental observation, going back to Lawvere [1963], allows us to consider Σ-theories as presentations of finitary monads. It will be instrumental in our study of syntactic presentations of generic automata, e.g. our Kleene Theorem (Theorem 4.13).

Definition 3.5 (*Presentation of a monad*). A Σ-theory $\mathcal{E}$ is said to be a *presentation* of the monad $\mathbf{T}$ if $\mathbf{T}$ is naturally isomorphic to $\mathbf{T}_\mathcal{E}$. We also say that $\mathcal{E}$ *generates* $\mathbf{T}$.

While the $\Sigma_\mathbf{T}$-theory $\mathcal{E}_\mathbf{T}$ yields a canonical presentation of the monad $\mathbf{T}$ we shall subsequently be interested in working out more compact presentations. In order to do this we will consider semantics of Σ-terms and Σ-theories over monads not necessarily of the form $\mathbf{T}_\mathcal{E}$. We will make free use of the equivalence between *n-ary algebraic operations* over a monad $\mathbf{T}$ and the elements of Tn (where we identify n with the set $\{1, \dots, n\}$). This equivalence was presented by Plotkin and Power [2003] (more generally as a duality between algebraic operations $n \to m$ and Kleisli morphisms $m \to Tn$), and we recall it below.

Let $\mathbf{T}$ be any monad, and recall that an *n-ary algebraic operation over* $\mathbf{T}$ is a natural transformation $\alpha : T^n \to T$, where T^n denotes the n-fold product $T \times \cdots \times T$,[1] such that for every $f : X \to TY$ we have

$$\begin{array}{ccc} (TX)^n & \xrightarrow{\alpha_X} & TX \\ {\scriptstyle (f^\star)^n}\downarrow & & \downarrow{\scriptstyle f^\star} \\ (TY)^n & \xrightarrow[\alpha_Y]{} & TY \end{array} \tag{3.1}$$

Any element $a \in Tn$ yields $\alpha : T^n \to T$ by defining

$$\alpha_X(f) = f^\star(a) = \mathsf{do}\ x \leftarrow a; f(x)$$

for any $f : n \to TX$. And given an n-ary algebraic operation $\alpha : T^n \to T$ over $\mathbf{T}$ we obtain $\alpha_n(\eta_n) \in Tn$. It is not difficult to show that these two passages are mutually inverse.

The technical advantage of using elements of Tn is that they are unconstrained whereas n-ary algebraic operations $\alpha : T^n \to T$ need to satisfy the above coherence condition (3.1).

Definition 3.6. Let Σ be a signature and let $\mathbf{T}$ be a (not necessarily finitary) monad. A *semantics* of Σ over $\mathbf{T}$ is an assignment $(\![-]\!)_T$ sending any $f : n \to 1$ in Σ to $(\![f]\!)_T \in Tn$. For every Σ-term t over a set of variables X this determines $(\![t]\!)_{TX} \in TX$ inductively as follows:

[1] We will use exponents on T only in this sense and *not* to indicate n-fold composition of T with itself.

— $(\!|x|\!)_{TX} = \eta_X(x)$ for $x \in X$;
— $(\!|f(t_1, \ldots, t_n)|\!)_{TX} = \mathsf{do}\ i \leftarrow (\!|f|\!)_T; (\!|t_i|\!)_{TX} = \alpha_X((\!|t_1|\!)_{TX}, \ldots, (\!|t_n|\!)_{TX})$, where $\alpha : T^n \to T$ is the n-ary algebraic operation over $\mathbf{T}$ corresponding to $(\!|f|\!)_T$.

Now let $\mathcal{E}$ be a Σ-theory. We call a semantics of Σ over $\mathbf{T}$

— *sound* if for any equation $s = t$ from $\mathcal{E}$ such that the free variables for s, t are included in X, $(\!|s|\!)_{TX} = (\!|t|\!)_{TX}$;
— *complete* if $s = t \in \mathcal{E}$ whenever $(\!|s|\!)_{TX} = (\!|t|\!)_{TX}$ for some X containing all the free variables of s and t;
— *expressive* if for every $p \in TX$ there is a Σ-term t over X such that $p = (\!|t|\!)_{TX}$.

In the future we shall omit the subscripts of $(\!|-|\!)$ whenever T or TX, respectively, are clear from the context. If a semantics of $\mathcal{E}$ over $\mathbf{T}$ is assumed, we simply call $\mathcal{E}$ *sound*, *complete* and *expressive* over $\mathbf{T}$ in the corresponding cases.

Remark 3.7. Note that if a Σ-theory is presented by a signature and axioms then it suffices to verify soundness for every axiom. Soundness of all equations in the closure $\mathcal{E}$ of the set of axioms under inference of standard equational logic then follows easily by induction.

Example 3.8. For every Σ-theory $\mathcal{E}$ we have a *canonical semantics* over the monad $\mathbf{T}_\mathcal{E}$ given by setting $(\!|f|\!) = f(1, 2, \ldots, n) \in T_\mathcal{E} n$.

The following theorem shows that the fact that a Σ-theory $\mathcal{E}$ generates a monad $\mathbf{T}$ entails a canonical presentation of $\mathbf{T}$ in terms of $\mathcal{E}$ up to isomorphism.

THEOREM 3.9. *Let $\mathcal{E}$ be a Σ-theory and let $\mathbf{T}$ be a finitary monad. Then $\mathcal{E}$ generates $\mathbf{T}$ iff there exists a sound, complete and expressive semantics of Σ over $\mathbf{T}$.*

PROOF. As we outlined after Definition 3.4, from $\mathcal{E}$ we can construct a finitary monad $\mathbf{T}_\mathcal{E}$ such that $T_\mathcal{E}X$ consists of Σ-terms over X modulo $\mathcal{E}$ and equip it with the canonical semantics $(\!|-|\!)$. Essentially due to Lawvere [1963] this semantics is sound, complete and expressive. Thus if $\mathcal{E}$ generates $\mathbf{T}$, i.e. we have a natural isomorphism $\gamma : \mathbf{T}_\mathcal{E} \to \mathbf{T}$, then we can define the semantics $(\!|-|\!)_\mathbf{T} = \gamma_n \cdot (\!|-|\!)$, and show by an easy induction that

$$(\!|-|\!)_{TX} = \gamma_X \cdot (\!|-|\!)_{T_\mathcal{E}X} \qquad \text{for every set } X. \tag{3.2}$$

Soundness, completeness and expressivity now easily follow from the fact that γ_X is bijective.

Conversely, we have to show that for any sound, complete and expressive semantics $(\!|-|\!)_\mathbf{T}$ of $\mathcal{E}$ over $\mathbf{T}$, the latter is isomorphic to $\mathbf{T}_\mathcal{E}$ via some natural isomorphism γ. Indeed, since any element of $T_\mathcal{E}X$ is represented by a Σ-term t we can define $\gamma_X : T_\mathcal{E}X \to TX$ by sending t to $(\!|t|\!)_{TX}$. It immediately follows by soundness that this definition is well-defined (i.e. independent of the concrete choice of t). Completeness and expressiveness of the given semantics imply injectivity and surjectivity, respectively, of γ_X. It is also easy to see by definition that γ_X respects unit and Kleisli lifting, hence it extends to a monad isomorphism. □

Example 3.10 (*Monads, Σ-theories*). Standard examples of computationally relevant monads include (cf. [Moggi 1991]) the following ones.

1. The *finite and unbounded powerset monads* $\mathcal{P}_\omega$ and $\mathcal{P}$. For both monads the unit is the singleton map $\eta_X : x \mapsto \{x\}$ and the Kleisli-lifting extends a map $f : X \to \mathcal{P}Y$ to $f^\star : \mathcal{P}X \to \mathcal{P}Y$ taking direct images: $f^\star(M \subseteq X) = \bigcup_{x \in M} f(x)$ (and similarly for $\mathcal{P}_\omega$). Only $\mathcal{P}_\omega$ is finitary and corresponds to the Σ-theory of join-semilattices with bottom over $\Sigma = \{\bot, \vee\}$, or equivalently to the theory of commutative idempotent monoids.
2. The *monoid action monad* for a monoid $(M, \cdot, 1)$ maps a set X to $M \times X$. Its unit is formed by the maps $\eta_X : x \mapsto (1, x)$ and the Kleisli-lifting extends $f : X \to M \times Y$ to $f^\star : M \times X \to M \times Y$

with $f^\star(m,x) = (m \cdot n, y)$ where $(n,y) = f(x)$. The corresponding Σ-theory is the theory of M-actions, i.e., Σ has a unary operation symbol $m \cdot (-)$ for every $m \in M$ with the usual axioms $m \cdot (n \cdot x) = (m \cdot n) \cdot x$ and $1 \cdot x = x$.

3. The *store monad* over a store S. The object assignment of this monad is $X \mapsto (X \times S)^S$ and the unit $\eta_X : X \to (X \times S)^S$ assigns $\eta_X(x) = \lambda s.\langle x, s\rangle$. Typically, S is the set of maps $L \to V$ from locations L to values V. A function $f : X \to (Y \times S)^S$ represents a computation that takes a value in X and, depending on the current contents of the store S returns a value in Y and a new store content. The Kleisli lifting sends f to $f^\star : (X \times S)^S \to (Y \times S)^S$ with

$$f^\star(h) = (S \xrightarrow{h} X \times S \xrightarrow{f \times S} (Y \times S)^S \times S \xrightarrow{\mathsf{ev}} Y \times S),$$

where ev is the obvious evaluation map. As shown in [Power and Shkaravska 2004], if V is finite then the corresponding store monad can be presented by a Σ-theory for $\Sigma = \{lookup_l : |V| \to 1\}_{l \in L} \cup \{update_{l,v} : 1 \to 1\}_{l \in L, v \in V}$.

4. The *continuation monad*. Given any set R, the assignment $X \mapsto R^{R^X}$ yields a monad under the following definitions:

$$\eta_X(x) = \lambda f.\, f(x) \qquad \text{and} \qquad f^\star(k) = \lambda c.\, k(\lambda x.\, f(x)(c)).$$

This monad is known to be non-finitary, unless $R = 1$.

We will need the following technical lemma for monads on $\mathbf{Set}$ and specifically implications from it for submonads of the store monad.

LEMMA 3.11. *Let T' be a submonad of T and let $\alpha : \mathsf{T} \to \mathsf{P}$ be a monad morphism. Then α restricted to T' induces a monad morphism $\alpha' : \mathsf{T}' \to \mathsf{P}'$ such that*

$$\begin{array}{ccc} \mathsf{T}' & \xrightarrow{\alpha'} & \mathsf{P}' \\ {\scriptstyle i}\downarrow & & \downarrow{\scriptstyle j} \\ \mathsf{T} & \xrightarrow{\alpha} & \mathsf{P} \end{array} \tag{3.3}$$

PROOF. For every set X take the factorization of $\alpha_X \cdot i_X$ into a surjective map $\alpha'_X : T'X \to P'X$ followed by an injective map (inclusion) $j_X : P'X \to PX$. Using the diagonal fill-in property of image factorizations, it is easy to verify that α' and j form natural transformations. Define $\eta'_X : X \to P'X$ as the composition of $\eta_X : X \to T'X$ and $\alpha'_X : T'X \to P'X$ and $\mu'_X : P'P'X \to P'X$ as the unique diagonal fill-in below (here $*$ denotes the usual horizontal composition of natural transformations):

$$\begin{array}{ccc} T'T'X & \xrightarrow{(\alpha' * \alpha')_X} & P'P'X \\ {\scriptstyle \alpha'_X \cdot \mu_X}\downarrow & \overset{\mu'_X}{\swarrow} & \downarrow{\scriptstyle \mu^P_X \cdot (j*j)_X} \\ P'X & \xrightarrow[j_X]{} & PX \end{array}$$

Indeed, $(\alpha' * \alpha')_X = T'\alpha'_X \cdot \alpha'_{P'X}$ is surjective since T' preserves surjections, and the outside square clearly commutes (using that $\alpha \cdot i$ is a monad morphism):

$$\mu^P_X \cdot (j * j)_X \cdot (\alpha' * \alpha')_X = \mu^P_X \cdot ((\alpha \cdot i) * (\alpha \dot{i}))_X = (\alpha \cdot i)_X \cdot \mu_X = j_X \cdot \alpha'_X \cdot \mu_X.$$

Using the unique diagonal fill-in property, it is now an easy exercise to verify that η' and μ' are natural, that (P', η', μ') satisfies the monad laws and that α' and j are monad morphisms. □

COROLLARY 3.12. *Let T_S be the store monad over S and let R_S be the reader monad over S (i.e. $R_S X = X^S$). For any submonad T of T_S, the monad morphism α sending any $f : S \to X \times S$ to $\pi_1 f : S \to X$ restricts to a submonad R of R_S.*

The following class of examples is especially relevant for the coalgebraic modelling of state-based systems.

Definition 3.13 (*Semimodule monad, Semimodule theory*). Given a semiring R, the semimodule monad $\mathbf{T}_R$ assigns to a set X the free left R-semimodule $\langle X\rangle_R$ over X. Explicitly, $\langle X\rangle_R$ consists of all formal linear combinations of the form

$$r_1 \cdot x_1 + \ldots + r_n \cdot x_n \qquad (r_i \in R, x_i \in X) \tag{3.4}$$

Equivalently, the elements of $\langle X\rangle_R$ are maps $f : X \to R$ with finite support (i.e. with $|\{x \in X \mid f(x) \neq 0\}| < \omega$). The assignment $X \mapsto \langle X\rangle_R$ extends to a monad, which we call the *(free) semimodule monad*: η_X sends any $x \in X$ to $1 \cdot x$ and $\theta^\star(p)$ applies the substitution $\theta : X \to \langle Y\rangle_R$ to $p \in \langle X\rangle_R$ and renormalizes the result as expected.

The semimodule monad corresponds to the Σ-theory of R-semimodules. Explicitly, we have a constant $\emptyset : 0 \to 1$, a binary operation $+ : 2 \to 1$, and a unary operation $\bar{r} : 1 \to 1$ for each $r \in R$. The axioms presenting this theory are the laws of commutative monoids for $+$ and $\emptyset$, plus the following identities for the (left) semiring action of R:

$$\bar{r}(x+y) = \bar{r}(x) + \bar{r}(y) \qquad \bar{r}(x) + \bar{s}(x) = \overline{r+s}(x) \qquad \bar{r}(\bar{s}(x)) = \overline{r \cdot s}(x)$$
$$\bar{r}(\emptyset) = \emptyset \qquad \bar{0}(x) = \emptyset \qquad \bar{1}(x) = x$$

It can be shown by using these laws that any term can by normalized to a term of the form $\bar{r}_1(x_1) + \ldots + \bar{r}_n(x_n)$, and the latter represent precisely the element (3.4) of $\langle X\rangle_R$. Thus, the above Σ-theory generates $\mathbf{T}_R$.

Some notable instances of $\mathbf{T}_R$ are the following:

— If R is the Boolean semiring $\{0, 1\}$ then $\mathbf{T}_R$ is (isomorphic to) the finite powerset monad $\mathcal{P}_\omega$.
— If R is the semiring of natural numbers then $\mathbf{T}_R$ is the *multiset* monad: the elements of $\langle X\rangle_R$ are in bijective correspondence with finite multisets over X.
— If R is the interval $[0, +\infty)$ then $\mathbf{T}_R$ is the monad of *finite valuations* used for modelling probabilistic computations [Varacca and Winskel 2006]. Two other well-known monads of *finite distributions* and *finite subdistributions* serving the same purpose embed into $\mathbf{T}_R$: the formal sums (3.4) for them are requested to satisfy the additional constraints $r_1 + \ldots + r_n = 1$ and $r_1 + \ldots + r_n \leq 1$, respectively.

3.1. The Stack Monad

The following example shows how to model a push-down store, see [Goncharov 2013].

Definition 3.14 (*Stack monad, Stack theory*). Given a finite set of stack symbols Γ, the *stack monad (over Γ)* is the submonad $\mathbf{T}$ of the store monad $(- \times \Gamma^*)^{\Gamma^*}$ for which the elements $\langle r, t\rangle$ of $TX \subseteq (X \times \Gamma^*)^{\Gamma^*}$ satisfy the following restriction: there exists k depending on r, t such that for every $w \in \Gamma^k$ and $u \in \Gamma^*$,

$$r(wu) = r(w) \qquad \text{and} \qquad t(wu) = t(w)u. \tag{3.5}$$

Intuitively, a function $f : X \to TY$ (cf. Example 3.10) has to compute its output in Y and result stack in Γ^* using only a portion of the stack of a predeclared size k that does not depend on the current content of the stack.

The *stack signature* w.r.t. $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ consists of operations $pop : n + 1 \to 1$ and $push_i : 1 \to 1$, $1 \leq i \leq n$. The intuition behind these operations is as follows (in each case the arguments represent continuations, i.e. computations that will be performed once the operation has completed its task, cf. [Plotkin and Power 2002]):

— $pop(x_1, \ldots, x_n, y)$ proceeds with y if the stack is empty; otherwise it removes the top element from it and proceeds with x_i, where $\gamma_i \in \Gamma$ is the removed stack element.
— $push_i(x)$ adds $\gamma_i \in \Gamma$ on top of the stack and proceeds with x.

(push-pop) $push_i(pop(x_1, \dots, x_n, y)) = x_i$

(pop-push) $pop(push_1(x), \dots, push_n(x), x) = x$

(pop-pop) $pop(x_1, \dots, x_n, pop(y_1, \dots, y_n, z)) = pop(x_1, \dots, x_n, z)$

Fig. 2. Axioms for the stack monad ($i \in \{1, \dots, n\}$).

The *stack theory* is presented by these operations and the axioms in Fig. 2. These axioms capture semantic equivalences of terms considered as programs transforming the underlying store. This implies that composition is to be read from left to right, e.g. the left-hand term of the first equation means "push γ_i, then pop one symbol from the stack, then proceed with with y if the stack was empty or with x_j if the popped symbol was γ_j. We connect the stack theory with the stack monad $\mathbf{T}$ by the following semantics:

$$(\!|pop|\!)(\epsilon) = \langle n+1, \epsilon\rangle, \qquad (\!|pop|\!)(\gamma_i w) = \langle i, w\rangle, \qquad (\!|push_i|\!)(w) = \langle 1, \gamma_i w\rangle$$

where $w \in \Gamma^*$, and ϵ denotes the empty stack.

As claimed in [Goncharov 2013] the stack theory generates the stack monad. We include a proof of this fact below. It relies on the following auxiliary statement.

LEMMA 3.15. *Any semantic identity*

$$(\!|pop(p_1, \dots, p_n, p)|\!) = (\!|pop(q_1, \dots, q_n, q)|\!)$$

with p and q not containing pop implies the semantic identities

$$(\!|p_1|\!) = (\!|q_1|\!), \dots, (\!|p_n|\!) = (\!|q_n|\!), (\!|p|\!) = (\!|q|\!).$$

PROOF. Using the semantics of pop, for any $1 \le i \le n$ and any $w \in \Gamma^*$,

$$(\!|p_i|\!)(w) = (\!|pop(p_1, \dots, p_n, p)|\!)(\gamma_i w) = (\!|pop(q_1, \dots, q_n, q)|\!)(\gamma_i w) = (\!|q_i|\!)(w).$$

Analogously, one proves $(\!|p|\!)(\epsilon) = (\!|q|\!)(\epsilon)$.

In order to prove $(\!|p|\!)(w) = (\!|q|\!)(w)$ for all words $w \in \Gamma^*$, we use that neither p nor q contain pop, i.e. both of them are nested applications of $push$ (with various indices) to some variables. Using the above semantics it is easy to calculate that for any $w \in \Gamma^*$,

$$(\!|p|\!)(w) = \langle x_1, u_1 w\rangle \qquad \text{and} \qquad (\!|q|\!)(w) = \langle x_2, u_2 w\rangle$$

for some $x_1, x_2 \in X$ and $u_1, u_2 \in \Gamma^*$ that do not depend on w. By substituting w with ϵ and using $(\!|p|\!)(\epsilon) = (\!|q|\!)(\epsilon)$ we obtain $x_1 = x_2$ and $u_1 = u_2$. It follows that $(\!|p|\!)(w) = (\!|q|\!)(w)$ holds for all $w \in \Gamma^*$ as desired. □

THEOREM 3.16. *The stack theory generates the stack monad.*

PROOF. We directly verify soundness, expressiveness and completeness in order.

— *Soundness* is straightforward to verify. Consider for example the left-hand side of the second axiom of the stack theory:

$$(\!|pop(push_1(x), \dots, push_n(x), x)|\!) = \mathsf{do}\ i \leftarrow (\!|pop|\!); \mathsf{if}\ (i < n+1)\ \mathsf{then}\ (\!|push_i(x)|\!)\ \mathsf{else}\ (\!|x|\!).$$

Using the definition of the store monad, and the semantic of $push$ and pop,

$$(\!|pop(push_1(x), \dots, push_n(x), x)|\!)(\epsilon) = (\!|x|\!)(\epsilon)$$
$$(\!|pop(push_1(x), \dots, push_n(x), x)|\!)(\gamma_i w) = (\!|push_i(x)|\!)(w) = (\!|x|\!)(\gamma_i w)$$

which is in agreement with the right-hand side of the identity in question.

— *Expressiveness.* Let $\langle r, t\rangle \in TX$. By definition, there is k such that for any $w \in \Gamma^k$ and any $u \in \Gamma^*$, (3.5) is satisfied. Using these data we construct by induction over k a Σ-term $p_k(r, t)$ over X:

— if $k = 0$ then $p_k(r,t) = push_{i_m}(\ldots push_{i_1}(r(\epsilon))\ldots)$ where $t(\epsilon) = \gamma_{i_1}\ldots\gamma_{i_m}$;
— if $k > 0$ let us define for any $1 \leq i \leq n$, $\langle r_i, t_i\rangle \in TX$ by the following equations

$$r_i(w) = r(\gamma_i w) \qquad \text{and} \qquad t_i(w) = t(\gamma_i w) \qquad \text{for every } w \in \Gamma^*.$$

Then we put $p_k(r,t) = pop(p_{k-1}(r_1,t_1),\ldots,p_{k-1}(r_n,t_n),p_0(r,t))$.
We now prove that $(\!|p_k(r,t)|\!) = \langle r,t\rangle$ by induction over k. For the base case $k = 0$, (3.5) states that for all $w \in \Gamma^*$ we have $r(w) = r(\epsilon)$ and $t(w) = t(\epsilon)w$. Therefore we have

$$\begin{aligned}(\!|p_0(r,t)|\!)(w) &= (\!|push_{i_m}(\ldots push_{i_1}(r(\epsilon))\ldots)|\!)(w)\\ &= \langle r(\epsilon), \gamma_{i_1}\cdots\gamma_{i_m}w\rangle && \text{// def. of } (\!|push_i|\!)\\ &= \langle r(w), t(w)\rangle && \text{// (3.5)}\end{aligned}$$

For the induction step note first that we may apply the induction hypothesis with r_i, t_i since this pair satisfies (3.5) for every $w \in \Gamma^{k-1}$. Thus we have

$$(\!|p_k(r,t)|\!)(\epsilon) = (\!|pop(p_{k-1}(r_1,t_1),\ldots,p_{k-1}(r_n,t_n),p_0(r,t))|\!)(\epsilon) = (\!|p_0(r,t)|\!)(\epsilon) = \langle r,t\rangle(\epsilon),$$

and furthermore we have

$$\begin{aligned}(\!|p_k(r,t)|\!)(\gamma_i w) &= (\!|pop(p_{k-1}(r_1,t_1),\ldots,p_{k-1}(r_n,t_n),p_0(r,t))|\!)(\gamma_i w)\\ &= (\!|p_{k-1}(r_i,t_i)|\!)(w) && \text{// def. of } (\!|pop|\!)\\ &= \langle r_i,t_i\rangle(w) && \text{// induction hypothesis}\\ &= \langle r,t\rangle(\gamma_i w).\end{aligned}$$

— *Completeness.* We turn the stack axioms into a rewriting system by orienting each equation from left to right. This rewriting system is obviously strongly normalizing because each application of the rule decreases the term size. There are no nontrivial critical pairs and therefore using the standard argument from term rewriting any term has a unique normal form [Terese 2003]. From the structure of the rules we can see that any normal form p either does not contain pop or is of the form $pop(p_1,\ldots,p_n,p')$ where each p_i is in a normal form and p' does not contain pop.

By soundness, it remains to show that for any normal p and q, $(\!|p|\!) = (\!|q|\!)$ implies $p = q \in \mathcal{E}$. We proceed by induction over the total number of the pop operators in p and q.

1. If both p and q do not contain pop they must be of the form $push_{i_1}(\ldots push_{i_m}(x)\ldots)$ and $push_{j_1}(\ldots push_{j_l}(y)\ldots)$, respectively. Then $(\!|p|\!)(w) = (\!|q|\!)(w)$ amounts to $\langle x, \gamma_{i_m}\ldots\gamma_{i_1}w\rangle = \langle y, \gamma_{j_l}\ldots\gamma_{i_1}w\rangle$ and therefore $x = y$, $m = l$ and $i_1 = j_1,\ldots,i_m = j_m$, i.e. p is identical to q.
2. If $p = pop(p_1,\ldots,p_n,p')$ and q does not contain pop, then we have

$$(\!|p|\!) = (\!|q|\!) = (\!|pop(push_1(q),\ldots,push_n(q),q)|\!).$$

By Lemma 3.15, $(\!|p_1|\!) = (\!|push_1(q)|\!),\ldots,(\!|p_n|\!) = (\!|push_n(q)|\!)$, $(\!|p'|\!) = (\!|q|\!)$. Note that the terms $push_i(q)$ need not be normal, but they can be normalized and since normalization only decreases the number of the pop operators the induction hypothesis applies to the result, and we have $\{p_1 = push_1(q),\ldots,p_n = push_n(q),p' = q\} \subseteq \mathcal{E}$. Hence, in $\mathcal{E}$, $p = pop(p_1,\ldots,p_n,p') = pop(push_1(q),\ldots,push_n(q),q) = q$.
3. If $q = pop(q_1,\ldots,q_n,q')$ and p does not contain pop, then we proceed analogously to the previous case.
4. If $p = pop(p_1,\ldots,p_n,p')$ and $q = pop(q_1,\ldots,q_n,q')$, then $(\!|p_i|\!) = (\!|q_i|\!)$, $i = 1,\ldots,n$, and $(\!|p'|\!) = (\!|q'|\!)$ by Lemma 3.15. By induction hypothesis, we have $\{p_1 = q_1,\ldots,p_n = q_n,p' = q'\} \subseteq \mathcal{E}$. Hence, in $\mathcal{E}$, $p = pop(p_1,\ldots,p_n,p') = pop(q_1,\ldots,q_n,q') = q$. □

3.2. The Tape Monad

We now introduce a monad and the corresponding theory underlying the tape of a Turing machine. The idea we use here is the same as in the case of the stack theory: we specify a submonad of a suitable store monad in such a way that only local transformations of the Turing tape are allowed.

Locality conditions:

$$t(i,\rho') \equiv t(i,\rho) \pmod{[i-k,i+k]} \qquad t(i,\rho) \equiv \rho \pmod{\overline{[i-k,i+k]}}$$

$$z(i,\rho') = z(i,\rho) \qquad |z(i,\rho) - i| \leq k \qquad r(i,\rho') = r(i,\rho)$$

Shift-invariance conditions:

$$t(i,\rho_{+j}) = t(i+j,\rho)_{+j} \qquad z(i,\rho_{+j}) = z(i+j,\rho) - j \qquad r(i,\rho_{+j}) = r(i+j,\rho)$$

Fig. 3. Conditions of the tape monad, assuming $\rho \equiv \rho' \pmod{[i-k,i+k]}$.

Let $\mathbb{Z}$ be the set of integers. We will need the following notation: given two maps $\rho, \rho' : \mathbb{Z} \to \Gamma$ and a set $I \subseteq \mathbb{Z}$ we write

$$\rho \equiv \rho' \pmod{I} \tag{3.6}$$

if $\rho(i) = \rho'(i)$ for all $i \in I$. We use interval notation to specify subsets of $\mathbb{Z}$, e.g.

$$[i-k,i+k] = \{j \mid i-k \leq j \leq i+k\},$$

and by $\overline{I}$ denote the complement of $I \subseteq \mathbb{Z}$. Also, for any $\rho : \mathbb{Z} \to \Gamma$ and any i, let $\rho_{+i} : \mathbb{Z} \to \Gamma$ be such that $\rho_{+i}(j) = \rho(i+j)$. The intuition here is that the maps ρ and ρ' represent snapshots of a Turing tape being filled with symbols from Γ (Γ may contain a special symbol for a blank cell, but it does not play a role sofar). The relation (3.6) indicates that ρ and ρ' agree on the positions indexed by I. The tape ρ_{+i} is obtained from ρ by reindexing the cells with the function $\lambda x.\, x - i$. We also commonly use the notation $\rho[k \mapsto \gamma_i]$ to refer to $\rho' : \mathbb{Z} \to \Gamma$ defined by $\rho'(k) = \gamma_i$ and $\rho'(l) = \rho(l)$ if $l \neq k$. This generalizes to sequences of assignments $k \mapsto \gamma_i$ in the obvious way.

Definition 3.17 (*Tape monad, Tape theory*). Let Γ be a finite set of tape symbols. The *tape monad (over* Γ*)* is the submonad $\mathbf{T}$ of the store monad $(\text{–} \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})^{\mathbb{Z} \times \Gamma^{\mathbb{Z}}}$ for which TX consists of exactly those maps

$$p = \langle r, z, t \rangle : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to (X \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})$$

satisfying the following restriction: there exists a $k \geq 0$, which we call a *locality parameter* of p, such that for any $i, j \in \mathbb{Z}$ and $\rho, \rho' : \mathbb{Z} \to \Gamma$ if $\rho \equiv \rho' \pmod{[i-k,i+k]}$ then the conditions in Fig. 3 are satisfied.

The *tape signature* w.r.t. $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ consists of the operations $rd : n \to 1$, $wr_i : 1 \to 1$ ($1 \leq i \leq n$), $mv_k : 1 \to 1$ ($k \in \{-1, 1\}$), which we interpret over any TX as follows:

$$\begin{aligned} (\!|rd|\!)(j,\rho) &= \langle i, j, \rho \rangle, && \text{where } \rho(j) = \gamma_i, \\ (\!|wr_i|\!)(j,\rho) &= \langle 1, j, \sigma[j \mapsto \gamma_i] \rangle, \\ (\!|mv_k|\!)(j,\rho) &= \langle 1, j+k, \rho \rangle. \end{aligned}$$

The *tape theory* w.r.t. Γ consist of all those equations $p = q$ in the tape signature, which are valid over every TX.

We shall henceforth use $mv_k(p)$ with arbitrary integer k as an abbreviation for p if $k = 0$; mv nested k times and applied to p if $k > 0$; and $mv_{\text{-}1}$ nested $-k$ times and applied to p if $k < 0$. It is straightforward to see that the semantic assignments remain intact under such extended use of the mv_k construct.

It is not obvious that Definition 3.17 does indeed define a monad. In order to show this, we will need the following auxiliary fact.

LEMMA 3.18. *Suppose, for some* $\rho, \rho', \theta, \theta' : \mathbb{Z} \to \Gamma$ *and* $I \subseteq J \subseteq \mathbb{Z}$ *that*

$$\rho \equiv \rho' \pmod{J}, \qquad \theta \equiv \theta' \pmod{I},$$
$$\rho \equiv \theta \pmod{\overline{I}}, \qquad \rho' \equiv \theta' \pmod{\overline{I}}.$$

Then $\theta \equiv \theta' \pmod{J}$.

PROOF. We need to prove that $\theta(i) = \theta'(i)$ for all $i \in J$. If $J = \emptyset$, we are done. So let $i \in J$. If $i \in I$, we are done since $\theta \equiv \theta' \pmod{I}$. Otherwise we have $i \notin I$ and obtain

$$\theta(i) = \rho(i) = \rho'(i) = \theta'(i)$$

by using the third, first, and last of the given equivalences. □

Now we can prove that Definition 3.17 correctly defines a monad.

THEOREM 3.19. *The conditions of Definition 3.17 identify a submonad* T *of the store monad over* $\mathbb{Z} \times \Gamma^{\mathbb{Z}}$.

PROOF. We have to show that the unit and Kleisli lifting of the store monad restrict to T. First recall the definition of the monad structure of the store monad over $\mathbb{Z} \times \Gamma^{\mathbb{Z}}$: for any $x \in X$, $f : X \to TY$ and $p = \langle r, z, t\rangle \in TX$,

$$\eta_X(x) : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to X \times \mathbb{Z} \times \Gamma^{\mathbb{Z}} \quad \text{with} \quad \eta_X(x)(i, \rho) = \langle x, i, \rho\rangle,$$
$$f^\star(p) : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to Y \times \mathbb{Z} \times \Gamma^{\mathbb{Z}} \quad \text{with} \quad f^\star(p)(i, \rho) = f(r(i, \rho))(z(i, \rho), t(i, \rho)).$$

It is our task to prove that the maps $\eta_X(x)$ and $f^\star(p)$ lie in TX and TY, respectively, i.e. they satisfy the conditions in Fig. 3. For $\eta_X(x)$ this clearly holds, for $\eta_X(x) = \langle r, z, t\rangle$, where z and t are the left- and right-hand product projections and r is the constant map on $x \in X$. We proceed to prove this for $f^\star(p)$.

Let $p \in TX$, let $f : X \to TY$ and let r, z, t, r_x, z_x, t_x be defined by

$$p(i, \rho) = \langle r(i, \rho), z(i, \rho), t(i, \rho)\rangle,$$
$$f(x)(i, \rho) = \langle r_x(i, \rho), z_x(i, \rho), t_x(i, \rho)\rangle$$

for any $x \in X$.

1. We first show the locality conditions for $f^\star(p)$. Let us fix a locality parameter k_p of $\langle r, z, t\rangle$. For any ρ and ρ' such that $\rho \equiv \rho' \pmod{[i - k_p, i + k_p]}$ we have that $r(i, \rho') = r(i, \rho)$ by the locality condition for r, and hence $f(r(i, \rho')) = f(r(i, \rho))$. Let k_f be a locality parameter of $f(r(i, \rho))$. Finally put $k = k_p + k_f$ and let us verify the locality conditions in Fig. 3 for $f^\star(p)$ using k as the corresponding locality parameter. First we calculate using the above notation:

$$\begin{aligned} f^\star(p)(i, \rho) &= f(r(i, \rho))(z(i, \rho), t(i, \rho)) \\ &= \langle r_{r(i,\rho)}(z(i, \rho), t(i, \rho)), z_{r(i,\rho)}(z(i, \rho), t(i, \rho)), t_{r(i,\rho)}(z(i, \rho), t(i, \rho))\rangle \\ &= \langle r_x(j, \theta), z_x(j, \theta), t_x(j, \theta)\rangle, \end{aligned}$$

where $x = r(i, \rho)$, $j = z(i, \rho)$ and $\theta = t(i, \rho)$ will be fixed from now on. Similarly,

$$\begin{aligned} f^\star(p)(i, \rho') &= f(r(i, \rho'))(z(i, \rho'), t(i, \rho')) \\ &= \langle r_{x'}(j', \theta'), z_{x'}(j', \theta'), t_{x'}(j', \theta')\rangle, \end{aligned}$$

where we also fix $x' = r(i, \rho')$, $j' = z(i, \rho')$ and $\theta' = t(i, \rho')$.

Let us fix such ρ, ρ' that $\rho \equiv \rho' \pmod{[i - k, i + k]}$. Note that this implies that

$$\rho \equiv \rho' \pmod{[i - k_p, i + k_p]} \qquad \text{and} \qquad \rho \equiv \rho' \pmod{[i - k_f, i + k_f]},$$

and therefore we can apply the locality conditions for both p (with locality parameter k_p) and for $f(r(i,\rho)) = f(r(i,\rho'))$ (with locality parameter k_f). The former immediately implies that

$$x = x' \qquad \text{and} \qquad j = j'. \tag{3.7}$$

On the other hand, using the locality condition for $f(r(i,\rho)) = f(r(i,\rho'))$ we obtain:

$$\begin{aligned} \theta' &\equiv \theta \pmod{[i-k_f, i+k_f]}, \\ \theta &\equiv \rho \pmod{\overline{[i-k_f, i+k_f]}}, \\ \theta' &\equiv \rho' \pmod{\overline{[i-k_f, i+k_f]}}. \end{aligned} \tag{3.8}$$

Hence, by Lemma 3.18, we conclude

$$\theta' \equiv \theta \pmod{[i-k, i+k]}. \tag{3.9}$$

Since $|i-j| \leq k_p$ and $k = k_p + k_f$, the interval $[i-k, i+k]$ includes $[j-k_f, j+k_f]$, hence (3.9) implies

$$\theta' \equiv \theta \pmod{[j-k_f, j+k_f]}. \tag{3.10}$$

We proceed to show the locality conditions for $f^\star(p)$.

— $t_x(j,\theta) \equiv t_{x'}(j',\theta') \pmod{[i-k, i+k]}$. By applying the locality conditions for $f(x) = f(r(i,\rho))$ to (3.10) we obtain

$$t_x(j,\theta') \equiv t_x(j,\theta) \pmod{[j-k_f, j+k_f]}, \tag{3.11}$$

$$t_x(j,\theta) \equiv \theta \pmod{\overline{[j-k_f, j+k_f]}}, \tag{3.12}$$

$$t_x(j,\theta') \equiv \theta' \pmod{\overline{[j-k_f, j+k_f]}}. \tag{3.13}$$

Recall that $|i-j| \leq k_p$ and $k = k_p + k_f$. Then by combining (3.12), (3.13) with (3.9) we obtain

$$t_x(j,\theta') \equiv t_x(j,\theta) \pmod{[i-k, j-k_f) \cup (j+k_f, i+k]}.$$

In conjunction with (3.11) and (3.7) this yields the desired result.

— $t_x(j,\theta) \equiv \rho \pmod{\overline{[i-k, i+k]}}$. Since $[i-k_f, i+k_f]$ is contained in $[i-k, i+k]$ this follows from (3.12) and (6.5),

— $z_x(j,\theta') = z_{x'}(j',\theta')$. By applying the locality condition for $f(x)$ to the assumption (3.10) we obtain $z_x(j,\theta) = z_x(j,\theta')$ and then we are done by (3.7).

— $|z_x(j,\theta) - i| \leq k$. We estimate the left-hand side as follows:

$$\begin{aligned} |z_x(j,\theta) - i| &= |z_x(j,\theta) - z(i,\rho) + z(i,\rho) - i| \\ &\leq |z_x(j,\theta) - z(i,\rho)| + |z(i,\rho) - i| \\ &= |z_x(j,\theta) - j| + |z(i,\rho) - i|. \end{aligned}$$

By applying the locality conditions for $f(x)$ and p to the assumptions $\rho \equiv \rho' \pmod{[i-k_p, i+k_p]}$ and (3.10), we see that the last sum is smaller or equal than $k_f + k_p = k$.

— $r_x(j,\theta') = r_{x'}(j',\theta')$. Using the locality conditions for $f(x)$ with (3.10) as assumption we obtain $r_x(j,\theta) = r_x(j,\theta')$ and hence $r_x(j,\theta) = r_{x'}(j',\theta')$ by (3.7).

2. We now prove the shift-invariance conditions for $f^\star(p)$. For this we need to compare $f^\star(p)(i,\rho_{+m})$ and $f^\star(p)(i+m,\rho)$ for any $i, m \in \mathbb{Z}$. Similarly as before let us define

$$\begin{aligned} x &= r(i,\rho_{+m}) & j &= z(i,\rho_{+m}) & \theta &= t(i,\rho_{+m}) \\ x' &= r(i+m,\rho) & j' &= z(i+m,\rho) & \theta' &= t(i+m,\rho) \end{aligned}$$

so that

$$\begin{aligned} f^\star(p)(i,\rho_{+m}) &= \langle r_x(j,\theta), z_x(j,\theta), t_x(j,\theta)\rangle, \\ f^\star(p)(i+m,\rho) &= \langle r_{x'}(j',\theta'), z_{x'}(j',\theta'), t_{x'}(j',\theta')\rangle. \end{aligned}$$

(mv-l) $mv_{\text{-}1}(mv_1(x)) = x$ **(rd-wr)** $rd(wr_1(x), \ldots, wr_n(x)) = x$

(mv-r) $mv_1(mv_{\text{-}1}(x)) = x$ **(wr-rd)** $wr_i(rd(x_1, \ldots, x_n)) = wr_i(x_i)$

(wr-wr) $wr_i(wr_j(x)) = wr_j(x)$

(wr-mv) $wr_i(mv_k(wr_j(mv_{\text{-}k}(x)))) = mv_k(wr_j(mv_{\text{-}k}(wr_i(x))))$

Fig. 4. Axioms for the tape monad ($k \in \mathbb{Z} \setminus \{0\}$, $i, j \in \{1, \ldots, n\}$).

Establishing the desired conditions now boils down to proving the following equations:

$$r_x(j, \theta) = r_{x'}(j', \theta'), \qquad z_x(j, \theta) = z_{x'}(j', \theta') - m, \qquad t_x(j, \theta) = t_{x'}(j', \theta')_{+m}.$$

The shift-invariance conditions of $p = \langle r, z, t \rangle$ state that

$$x = x', \qquad j + m = j', \qquad \theta = \theta'_{+m}.$$

Using these equations and the shift-invariance conditions of $f(x)$ we obtain the desired equations:

$$\begin{aligned} r_x(j, \theta) &= r_{x'}(j, \theta) = r_{x'}(j, \theta'_{+m}) = r_{x'}(j + m, \theta') = r_{x'}(j', \theta'), \\ z_x(j, \theta) &= z_{x'}(j, \theta) = z_{x'}(j, \theta'_{+m}) = z_{x'}(j + m, \theta') - m = z_{x'}(j', \theta') - m, \\ t_x(j, \theta) &= t_{x'}(j, \theta) = t_{x'}(j, \theta'_{+m}) = t_{x'}(j + m, \theta')_{+m} = t_{x'}(j', \theta')_{+m}. \end{aligned}$$

This completes the proof. □

In contrast to the stack theory, the tape theory is so far defined indirectly. We present the corresponding *infinitary* axiomatization for it in Fig. 4. Like in the case of the stack theory, these equations capture semantic equivalences of terms considered as programs transforming the underlying store. This implies that composition is to be read from left to right, e.g. $wr_i(wr_j(x))$ means "write γ_i, then γ_j, then proceed with x".

THEOREM 3.20. *The deductive closure of the axioms in Fig. 4 generates the tape monad over* $\Gamma = \{\gamma, \ldots, \gamma_n\}$.

Proving Theorem 3.20 requires some preliminaries. Let us introduce the following auxiliary operations: $wr_{i,k} : 1 \to 1$, $rd_k : n \to 1$ with k ranging over all integers and i ranging from 1 to n. These are just abbreviations for the following derived operations:

$$\begin{aligned} wr_{i,k}(x) &= mv_k(wr_i(mv_{\text{-}k}(x))) \\ rd_k(x_1, \ldots, x_n) &= mv_k(rd(mv_{\text{-}k}(x_1), \ldots, mv_{\text{-}k}(x_n))) \end{aligned}$$

Note that $wr_{i,0} = wr_i$, $rd_0 = rd$. Clearly, we have

$$\begin{aligned} (\!|wr_{i,k}|\!)_T(j, \rho) &= \langle 1, j, \rho[j + k \mapsto \gamma_i] \rangle, \\ (\!|rd_k|\!)_T(j, \rho[j + k \mapsto \gamma_i]) &= \langle i, j, \rho[j + k \mapsto \gamma_i] \rangle. \end{aligned}$$

It is easy to establish the following implications of the axioms in Fig. 4.

LEMMA 3.21. *The following proof rule is sound w.r.t. the axioms in Fig. 4:*

$$\frac{wr_{1,k}(s) = wr_{1,k}(t) \quad \cdots \quad wr_{n,k}(s) = wr_{n,k}(t)}{s = t} \qquad \textit{for every } k \in \mathbb{Z}.$$

PROOF. Indeed we have

$$\begin{aligned} s &= rd_k(wr_{1,k}(s), \ldots, wr_{n,k}(s)) && \text{// \textbf{(mv-l)}, \textbf{(mv-r)}, \textbf{(rd-wr)}} \\ &= rd_k(wr_{1,k}(t), \ldots, wr_{n,k}(t)) && \text{// premises} \\ &= t && \text{// \textbf{(mv-l)}, \textbf{(mv-r)}, \textbf{(rd-wr)}} \end{aligned}$$

□

LEMMA 3.22. *The following equations are derivable from the ones in Fig. 4.*

$$wr_{i,k}(wr_{j,k}(x)) = wr_{j,k}(x) \tag{3.14}$$
$$wr_{i,k}(wr_{j,k'}(x)) = wr_{j,k'}(wr_{i,k}(x)) \qquad (k \neq k') \tag{3.15}$$
$$wr_{i,k}(rd_k(r_1,\dots,r_n)) = wr_{i,k}(r_i) \tag{3.16}$$
$$wr_{i,k}(rd_{k'}(r_1,\dots,r_n)) = rd_{k'}(wr_{i,k}(r_1),\dots,wr_{i,k}(r_n)) \qquad (k \neq k') \tag{3.17}$$

PROOF. Equation (3.14) is shown as follows:

$$\begin{aligned} &wr_{i,k}(wr_{j,k}(x)) \\ &\quad = mv_k(wr_i(mv_{\text{-}k}(mv_k(wr_j(mv_{\text{-}k}(x)))))) && \text{// definition} \\ &\quad = mv_k(wr_i(wr_j(mv_{\text{-}k}(x)))) && \text{// \textbf{(mv-l)}, \textbf{(mv-r)}} \\ &\quad = mv_k(wr_j(mv_{\text{-}k}(x))) && \text{// \textbf{(wr-wr)}} \\ &\quad = wr_{j,k}(x). && \text{// definition} \end{aligned}$$

Analogously one obtains (3.16) using **(wr-rd)**. Let us show (3.15):

$$\begin{aligned} &wr_{i,k}(wr_{j,k'}(x)) \\ &\quad = mv_k(wr_i(mv_{\text{-}k}(mv_{k'}(wr_j(mv_{\text{-}k'}(x)))))) && \text{// definition} \\ &\quad = mv_k(wr_i(mv_{k'-k}(wr_j(mv_{\text{-}k'}(x))))) && \text{// \textbf{(mv-l)}, \textbf{(mv-r)}} \\ &\quad = mv_{k'}(mv_{k-k'}(wr_i(mv_{k'-k}(wr_j(mv_{\text{-}k'}(x)))))) && \text{// \textbf{(mv-l)}, \textbf{(mv-r)}} \\ &\quad = mv_{k'}(wr_j(mv_{k-k'}(wr_i(mv_{k'-k}(mv_{\text{-}k'}(x)))))) && \text{// \textbf{(wr-mv)}} \\ &\quad = mv_{k'}(wr_j(mv_{k-k'}(wr_i(mv_{\text{-}k}(x))))) && \text{// \textbf{(mv-l)}, \textbf{(mv-r)}} \\ &\quad = mv_{k'}(wr_j(mv_{\text{-}k'}(mv_k(wr_i(mv_{\text{-}k}(x)))))) && \text{// \textbf{(mv-l)}, \textbf{(mv-r)}} \\ &\quad = wr_{j,k'}(wr_{i,k}(x)). && \text{// definition} \end{aligned}$$

Finally, let us show (3.17). To this end, apply $wr_{j,k'}$ to both sides of the identity and simplify the result. For the left-hand side of the equation we obtain

$$\begin{aligned} &wr_{j,k'}(wr_{i,k}(rd_{k'}(r_1,\dots,r_n))) \\ &\quad = wr_{i,k}(wr_{j,k'}(rd_{k'}(r_1,\dots,r_n))) && \text{// (3.15)} \\ &\quad = wr_{i,k}(wr_{j,k'}(r_j)), && \text{// (3.16)} \end{aligned}$$

and for the right-hand side,

$$\begin{aligned} &wr_{j,k'}(rd_{k'}(wr_{i,k}(r_1),\dots,wr_{i,k}(r_n))) \\ &\quad = wr_{j,k'}(wr_{i,k}(r_j)) && \text{// (3.16)} \\ &\quad = wr_{i,k}(wr_{j,k'}(r_j)). && \text{// (3.15)} \end{aligned}$$

We are now done by Lemma 3.21, since the desired equation holds when $wr_{j,k'}$ is applied to both sides for every $k' \in \mathbb{Z}$. □

PROOF OF THEOREM 3.20. By Theorem 3.9 it suffices to verify the following.

— *Soundness.* This is a routine calculation using $(\!|-|\!)$ from Definition 3.6.
— *Expressiveness.* Recall that for any $p = \langle r, z, t\rangle : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to (X \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})$ in TX there exists k for which the conditions in Fig. 3 are satisfied. We claim that

$$\begin{aligned} p = \mathsf{do}\ & x_{\text{-}k} \leftarrow (\!|rd_{\text{-}k}|\!)_T; \dots; x_k \leftarrow (\!|rd_k|\!)_T; \\ & (\!|wr_{t(0,p(x_{\text{-}k},\dots,x_k))(\text{-}k),\text{-}k}|\!)_T; \dots; (\!|wr_{t(0,p(x_{\text{-}k},\dots,x_k))(k),k}|\!)_T; \\ & (\!|mv_{z(0,p(x_{\text{-}k},\dots,x_k))}|\!)_T; (\!|r(0,h(x_{\text{-}k},\dots,x_k))|\!) \end{aligned} \tag{3.18}$$

(slightly abusing the notation by writing $wr_{\gamma_i,j}$ in lieu of $wr_{i,j}$) where $h : \Gamma^{2k+1} \to \Gamma^{\mathbb{Z}}$ is any map for which $h(\gamma_{i_{-k}}, \dots, \gamma_{i_k})(j) = \gamma_{i_j}$ whenever $-k \le j \le k$. Intuitively, the constructed program works as follows: in the first step it reads values from the interval $[-k, k]$ on the tape relative to the current head position, and stores the obtained results in $x_{-k}, \dots, x_k$; in the step round it updated the tape according to t; in the third step, it moves the head according to z; and in the final fourth step, it returns the result from X computed by r.

Once we prove (3.18) we are done with the proof of expressiveness; indeed, recall from Definition 3.6 that for any $f : m \to 1$ and any family of terms $t_1, \dots t_m$,

$$(\!|x|\!) = \eta_X(x) \qquad (\!|f(t_1, \dots, t_m)|\!) = \mathsf{do}\ i \leftarrow (\!|f|\!)_T; (\!|t_i|\!).$$

Then a straightforward induction argument shows that the right-hand side of (3.18) is $(\!|t|\!)$ for some term t.

Now we prove (3.18). Let $j \in \mathbb{Z}$ and let $\rho : \mathbb{Z} \to \Gamma$. Applying the right-hand side of (3.18) to $\langle j, \rho\rangle$ and using the semantics of rd we obtain

$$\big(\mathsf{do}\ (\!|wr_{t(0,\theta)(-k),-k}|\!)_T; \dots; (\!|wr_{t(0,\theta)(-k),-k}|\!)_T; (\!|mv_{z(0,\theta)}|\!)_T; (\!|r(0,\theta)|\!)\big)\,\langle j, \rho\rangle$$

for some $\theta : \mathbb{Z} \to \Gamma$ such that

$$\rho_{+j} \equiv \theta \mod [-k, k]. \tag{3.19}$$

Using the semantics of wr we further reduce the right-hand side of (3.18) to

$$\big(\mathsf{do}\ (\!|mv_{z(0,\theta)}|\!)_T; (\!|r(0,\theta)|\!)\big)\,\langle j, \rho[j-k \mapsto t(0,\theta)(-k), \dots, j+k \mapsto t(0,\theta)(k)]\rangle$$

and the latter is equal to

$$\langle r(0,\theta), j + z(0,\theta), \rho[j-k \mapsto t(0,\theta)(-k), \dots, j+k \mapsto t(0,\theta)(k)]\rangle.$$

It remains to show that this is equal to $p(j,\rho) = \langle r(j,\rho), z(j,\rho), t(j,\rho)\rangle$. Consider the first component: using shift-invariance and the locality condition with assumption (3.19) for r we have $r(j,\rho) = r(0,\rho_{+j}) = r(0,\theta)$.

Analogously, for the second component, $z(j,\rho) = j + z(0,\rho_{+j}) = j + z(0,\theta)$.

Finally, consider the third component. We shall prove separately that

$$t(j,\rho) \equiv \rho[j-k \mapsto t(0,\theta)(-k), \dots, j+k \mapsto t(0,\theta)(k)] \mod [j-k, j+k]$$
$$t(j,\rho) \equiv \rho[j-k \mapsto t(0,\theta)(-k), \dots, j+k \mapsto t(0,\theta)(k)] \mod \overline{[j-k, j+k]}$$

The first congruence is equivalent to

$$\begin{aligned} t(j,\rho)_{+j} &\equiv \rho_{+j}[-k \mapsto t(0,\theta)(-k), \dots, k \mapsto t(0,\theta)(k)] \mod [-k,k] \\ &\equiv t(0,\theta) \mod [-k,k]. \end{aligned}$$

The last congruence clearly holds: by shift-invariance for t, $t(j,\rho)_{+j} = t(0,\rho_{+j})$ and by the first locality condition for t, $t(0,\rho_{+j}) \equiv t(0,\theta) \mod [-k,k]$ (using (3.19)).

Let us check the second congruence. It is equivalent to $t(j,\rho)_{+j} \equiv \rho_{+j} \mod \overline{[-k,k]}$. Like before $t(j,\rho)_{+j} = t(0,\rho_{+j})$ and using the second locality condition for t, we have $t(0,\rho_{+j}) \equiv \rho_{+j} \mod \overline{[-k,k]}$ which completes the proof of expressiveness.

— *Completeness.* Let s and t be tape theory terms such that $(\!|s|\!) = (\!|t|\!)$. We have to prove that $s = t$ is a provable identity. We first consider the case where both s and t are *normal*, that means that s and t are composed from rd_k, $wr_{i,k}$, mv_k and variables in such a way that (i) no rd occurs under wr; (ii) mv is only applied to variables. We proceed by induction over the total number of rd operations in $\langle s, t\rangle$.

In the base case neither s nor t contains rd and hence

$$s = wr_{k_1,i_1}(\dots(wr_{k_m,i_m}(mv_k(x)))\dots), \qquad t = wr_{l_1,j_1}(\dots(wr_{l_w,j_w}(mv_l(y)))\dots)$$

with suitable indices and variables x, y. W.l.o.g. we can assume that the sequences $k_1, \dots, k_m$ and $l_1, \dots, l_w$ are increasing and nonrepetetive – otherwise we can rearrange and possibly remove some of the wr operators by Lemma 3.22. Then we argue that s must be provably equal to t. We have for all v and ρ that $(\!|s|\!)(v, \rho) = \langle x, v + k, \rho_1 \rangle$ and $(\!|t|\!)(v, \rho) = \langle y, v + l, \rho_2 \rangle$ for some x, y, k, l, ρ_1 and ρ_2. By hypothesis we have $k = l$, $x = y$ and $\rho_1 = \rho_2$. Moreover, $\rho_1 = \rho[k_1 \mapsto \gamma_{i_1}, \dots, k_m \mapsto \gamma_{i_m}]$ and $\rho_2 = \rho[l_1 \mapsto \gamma_{j_1}, \dots, l_w \mapsto \gamma_{j_w}]$. Since these are equal for all ρ it follows that the sequences $\langle k_1, i_1 \rangle, \dots, \langle k_m, i_m \rangle$ and $\langle l_1, j_1 \rangle, \dots, \langle l_w, j_w \rangle$ must be equal, too.

For the induction step suppose that $s = rd_k(r_1, \dots, r_n)$. We apply $wr_{i,k}$ to s and t for every i. Note that any term $wr_{i,k}(s)$ can be transformed to a normal form s_i using (3.16) and (3.17) as rewrite rules as follows:

$$wr_{i,k}(rd_k(r_1, \dots, r_n)) \to wr_{i,k}(r_i) \tag{3.20}$$

$$wr_{i,k}(rd_{k'}(r_1, \dots, r_n)) \to rd_{k'}(wr_{i,k}(r_1), \dots, wr_{i,k}(r_n)) \tag{3.21}$$

where $k' \neq k$. In the same way any $wr_{i,k}(t)$ reduces to a normal form t_i. Since $(\!|s|\!) = (\!|t|\!)$ we have for every i that

$$(\!|s_i|\!) = (\!|wr_{i,k}(s)|\!) = (\!|wr_{i,k}(t)|\!) = (\!|t_i|\!).$$

Now notice that s_i has at least one rd operator less than s; thus, the total number of rd operators in $\langle s_i, t_i \rangle$ is lower than that of $\langle s, t \rangle$. Hence, by induction hypothesis,

$$wr_{i,k}(s) = s_i = t_i = wr_{i,k}(t)$$

are provable identies in the tape theory for every i. By Lemma 3.21, $s = t$ is a provable identity as desired.

The remaining case $t = rd_k(t_1, \dots, t_n)$ is symmetric to the previous one.

In order to complete the proof it remains to show how an arbitrary tape theory term t can be reduced to a normal form satisfying the above conditions (i) and (ii) in such a way that the reductions are sound w.r.t. the identities in Fig. 4. Given t we ensure first (ii) and then (i) as follows.

(ii) We exhaustively apply the reductions

$$mv_k(mv_l(s)) \to mv_{k+l}(s)$$
$$mv_k(wr_{i,l}(s)) \to wr_{i,k+l}(mv_k(s))$$
$$mv_k(rd_l(s_1, \dots, s_n)) \to rd_{k+l}(mv_k(s_1), \dots, mv_k(s_n))$$

which are easily seen to be sound by **(mv-l)** and **(mv-r)**.

(i) Then we exhaustively apply (3.20) and (3.21). □

It can now be readily shown that any axiomatization of the tape monad is necessarily infinitary.

THEOREM 3.23. *The tape theory over Γ is not finitely axiomatizable, unless $|\Gamma| \leq 1$.*

PROOF. If $|\Gamma| = 0$ then the axiom scheme **(wr-mv)** disappears instantly, and if $|\Gamma| = 1$ then it is entailed by the axioms **(mv-l)**, **(mv-r)** and the identity $wr(x) = x$ (we omit the index 1 at wr); this identity is derived as follows using **(rd-wr)** and **(wr-wr)**:

$$x = rd(wr(x)) = rd(wr(wr(x))) = wr(x).$$

Note that for $|\Gamma| = 0$, the monad becomes trivial ($TX = \emptyset$) and for $|\Gamma| = 1$, $TX = X \times \mathbb{Z}$, for it captures precisely those moves $z : \mathbb{Z} \to \mathbb{Z}$ of the head for which by the second shift-invariance condition, $z(i) = z(0) + i$, i.e. such transformations that are precisely characterized by a single number $z(0)$.

Let us assume henceforth that $|\Gamma| \geq 2$. Given a finite set of identities $\mathcal{A}$ belonging to the tape theory, we prove the claim by constructing a model M of $\mathcal{A}$ which does not satisfy all instances of **(wr-mv)**. Let m be greater than the total number of instances of operations mv_{-1} and mv_1 in any equation from $\mathcal{A}$. Our model M is carried by the set of all endomaps on a tape of length m, i.e. all endomaps on the set $\mathbb{Z}_m \times \Gamma^{\mathbb{Z}_m} \to \mathbb{Z}_m \times \Gamma^{\mathbb{Z}_m}$, where $\mathbb{Z}_m = \{0, \dots, m-1\}$ is the finite ring

of integers modulo m. We interpret the operations of the tape theory on M (here we overload our previous notation $(\!|-|\!)_{TX}$ and write $(\!|-|\!)_M$ for this interpretation) as follows:

$$\begin{aligned}(\!|rd|\!)_M(p_1,\dots,p_n)(z,\rho) &= p_i(z,\rho), \qquad \text{where } \rho(z)=\gamma_i,\\ (\!|wr_i|\!)_M(p)(z,\rho) &= p(z,\rho[z\mapsto\gamma_i])\\ (\!|mv_k|\!)_M(p)(z,\rho) &= p(z+_m k,\rho)\end{aligned}$$

where i ranges from 1 to $n = |\Gamma|$ and $+_m$ denotes addition modulo m. By additionally defining $(\!|x|\!)_M = \mathsf{id}$ for every variable x, we extend $(\!|-|\!)_M$ to terms over the tape signature. The inductive clauses for $(\!|p|\!)_M(z,\rho)$ are the same as for $(\!|p|\!)_{T1}(z,\rho)$, except that the tuples returned by the latter interpretation are extended to the left with an additional component constantly equal 1, and by the fact that $(\!|p|\!)_M(z,\rho)$ may call on addition modulo m for sufficiently large z and sufficiently many operations mv_k in p. Specifically, this means that for any equation $p = q$ in $\mathcal{A}$, any $z \in \mathbb{Z}$ and any $\theta : \mathbb{Z}_m \to \Gamma$,

$$(\!|p|\!)_M(0,\theta) = \langle z',\theta'\rangle \qquad \text{iff} \qquad (\!|p|\!)_{T1}(0,\theta_*) = \langle 1, z, \theta'_*\rangle$$

for $\theta_*, \theta'_* : \mathbb{Z} \to \Gamma$ defined as follows:

$$\theta_*(i) = \theta(i \bmod m) \qquad \text{and} \qquad \theta'_*(i) = \theta'(i \bmod m) \qquad \text{for every } i \in \mathbb{Z}.$$

An analogous identity holds for q and therefore

$$(\!|p|\!)_M(0,\theta) = (\!|q|\!)_M(0,\theta) \qquad \text{for every } \theta : \mathbb{Z}_m \to \Gamma. \tag{3.22}$$

Now note that, for any $z \in \mathbb{Z}_m$, $\theta \in \Gamma^{\mathbb{Z}_m}$, in order to compute $(\!|p|\!)_M$ on (z,θ) one can first perform a cyclic left-shift on the model, then apply $(\!|p|\!)_M$ with 0 it its first argument and then shift the result back to the right. More precisely, let $\theta_{+z}(i) = \theta(i +_m z)$ for every $i, z \in \mathbb{Z}$ and $\theta : \mathbb{Z} \to \Gamma$ (in analogy the same notation ρ_{+z} we previously used for $\rho : \mathbb{Z} \to \Gamma$). Then we have

$$(\!|p|\!)_M(z,\theta) = \langle z' +_m z, \theta'_{-z}\rangle, \qquad \text{where } \langle z',\theta'\rangle = (\!|p|\!)_M\langle 0,\theta_{+z}\rangle.$$

This can be shown by a straightforward induction over the term p.

Hence, from (3.22), we obtain that $(\!|p|\!)_M(z,\theta) = (\!|q|\!)_M(z,\theta)$ for every $\theta : \mathbb{Z}_m \to \Gamma$ and $z \in \mathbb{Z}_m$. We have thus shown that $\mathcal{A}$ is valid over M.

Now, if we take $k = m$ in **(wr-mv)** we obtain that for $i \neq j$ (such a pair of indices exists for $|\Gamma| \geq 2$, by assumption):

$$\begin{aligned}(\!|wr_i(mv_m(wr_j(mv_{\text{-}m}(x))))|\!)_M(0,\rho) &= \langle 0, \rho[0\mapsto\gamma_j]\rangle\\ &\neq \langle 0,\rho[0\mapsto\gamma_i]\rangle\\ &= (\!|mv_m(wr_j(mv_{\text{-}m}(wr_i(x))))|\!)_M(0,\rho)\end{aligned}$$

This concludes the proof. □

4. REACTIVE T-ALGEBRAS AND T-AUTOMATA

As in Section 2 we fix a finite set of actions A. We first consider $\mathbf{T}$-algebras which are equipped with a transition structure similar to that of Moore automata but which, in addition, preserves the algebraic structure. Such a transition structure extends a $\mathbf{T}$-algebra with dynamic behaviour (making it into a coalgebra) and hence we call such structures *reactive* $\mathbf{T}$-algebras.

Definition 4.1 (*Reactive* $\mathbf{T}$*-algebra*). Let B and X be $\mathbf{T}$-algebras. Then X is a *reactive* $\mathbf{T}$*-algebra* if X is an coalgebra for $L = B \times (-)^A$ (cf. Definition 2.1) for which $\partial_a : X \to X$ and $o : X \to B$ are $\mathbf{T}$-algebra morphisms.

Remark 4.2. The definition of a reactive $\mathbf{T}$-algebra is an instance of a more general construction [Bartels 2004] (the main idea goes back to Turi and Plotkin [1997]). Any endofunctor

$F : \mathbf{Set} \to \mathbf{Set}$ equipped with a distributive law $\delta : \mathsf{T}F \to F\mathsf{T}$ is known to lift to the Eilenberg-Moore category $\mathbf{Set}^{\mathsf{T}}$. Under $F = L$ there is a standard distributive law, given by

$$T(B \times X^A) \xrightarrow{\langle T\pi_0, T\pi_1\rangle} TB \times T(X^A) \xrightarrow{\alpha \times \langle T\mathsf{ev}_a\rangle_{a\in A}} B \times (TX)^A$$

where π_0, π_1 denote the product projections, $\alpha : TB \to B$ is the T-algebra structure on B, $\mathsf{ev}_a : X^A \to X$ is the obvious evaluation at $a \in A$, and we regard $(TX)^A$ as the $|A|$-fold power of TX. A reactive T-algebra is then simply a coalgebra in $\mathbf{Set}^{\mathsf{T}}$ for the lifting of L. Putting it yet differently, a reactive T-algebra is a *δ-bialgebra* for the above distributive law δ [Jacobs 2006] (see also [Klin 2011]).

Given a T-algebra B, the set of all formal power series B^{A^*} (which is the carrier of the final L-coalgebra in $\mathbf{Set}$) can also be viewed as a reactive T-algebra with a pointwise T-algebra structure. The morphisms ∂_a and o are easily seen to be T-algebra morphisms. Since every reactive T-algebra is an L-coalgebra, reactive T-algebras inherit the general coalgebraic theory from Section 2. In particular, we use for reactive T-algebras the same notions of language semantics and language equivalence as for L-coalgebras (see Definition 2.1).

Definition 4.3 (T*-automaton, cf. [Jacobs 2006]*). Suppose, T is finitary and B is finitely generated, i.e. there is a finite set B_0 of *generators* and a surjection $TB_0 \to B$ underlying a T-algebra morphism. A T*-automaton* m is given by a triple of maps

$$o^m : X \to B, \qquad t^m : A \times X \to TX, \qquad \alpha^m : TB \to B, \qquad (\bigstar)$$

where α^m is a T-algebra and X is finite. The first two maps in ($\bigstar$) can be aggregated into a coalgebra transition structure, which we write as

$$m : X \to B \times (TX)^A$$

slightly abusing the notation.

Remark 4.4. We require the monad T in ($\bigstar$) to be finitary in order to be able to represent T-automata using finite syntax. For technical reasons, it is sometimes convenient to drop this restriction (e.g. in Section 8 where T is the continuation monad). This is not in conflict with Definition 4.3, since we apply T to finite sets only, and therefore, in lieu of T, we can use its finitary coreflection T_ω whose object part is defined by $T_\omega X = \bigcup_{Y\subseteq X, |Y|<\omega} TY$.

A simple nontrivial example of a T-automaton is given with the *nondeterministic finite state machines (NFSM)* by taking $B = \{0, 1\}$, $\mathsf{T} = \mathcal{P}_\omega$ and $\alpha^m(s \subseteq \{0, 1\}) = 1$ iff $1 \in s$.

In order to introduce the language semantics of a T-automaton we will first convert it into a reactive T-algebra, and the language semantics of the latter is settled by Definition 2.1. This conversion is called the *generalized powerset construction* [Silva et al. 2013], as it generalizes the classical Rabin-Scott NFSM determinization [Rabin and Scott 1959] and amounts to the following. Observe that LTX is a T-algebra, since TX is the free T-algebra on X and L lifts to $\mathbf{Set}^{\mathsf{T}}$ (see Remark 4.2). Hence, given a T-automaton $m : X \to B \times (TX)^A$ there exists a unique T-algebra morphism

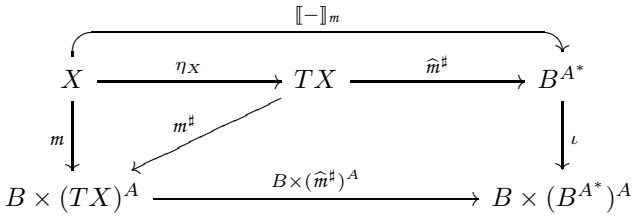
$$m^\sharp : TX \to B \times (TX)^A$$

such that $m^\sharp \cdot \eta_X = m$; explicitly, $m^\sharp(p) = (B \times \mu_X^A) \cdot \delta_{TX} \cdot Tm$ where δ is the distributive law from Remark 4.2. This $m^\sharp$ is a reactive T-algebra on TX.

Definition 4.5. Given a T-automaton $m : X \to B \times (TX)^A$, its *language semantics* assigns to every state $x \in X$ the formal power series

$$[\![x]\!]_m = \widehat{m^\sharp}(\eta_X(x)) : A^* \to B,$$

where $\widehat{m^\sharp}$ is the unique L-coalgebra morphism from $(TX, m^\sharp)$ to the final coalgebra (B^{A^*}, ι). This can be summarized in the following diagram



(4.1)

Remark 4.6.

(1) Due to the 1-1-correspondence of m and $m^\sharp$ given by freeness of TX, T-automata bijectively correspond to reactive T-algebras whose carrier is a free algebra on a finite set; but we find it useful to retain the distinction.

(2) The term language semantics comes from the fact that for $\mathsf{T} = \mathcal{P}_\omega$ and $B = \{0, 1\}$, our language semantics of T-automata is precisely the classical language semantics of NFSM; $[\![x]\!]_m$ is the formal language accepted by the NFSM given by m with initial state x.
More generally, for any semiring R, take $B = R$ and the semimodule monad T_R. Then T-automata are precisely weighted automata with weights in the semiring R, and for every state x the formal power-series $[\![x]\!]_m : A^* \to R$ is the weighted language accepted by the weighted automaton given by m.
However, for other monads T and algebras B elements in B^{A^*} may look very different than formal languages, e.g. for the stack T-automata we will discuss in Section 5.1.

(3) Note that T-automata for the identity monad are precisely the same as Moore automata, and the above definition of their language semantics coincides with Definition 2.1.

Note that the generalized powerset construction does not reduce a T-automaton to a Moore automaton over TX as TX need not be finite. However, when this is the case, e.g. for $\mathsf{T} = \mathcal{P}_\omega$, the semantics of a T-automaton falls within regular power series, which is precisely the reason why the languages recognized by deterministic and nondeterministic FSM coincide. Surprisingly, all T-automata with a finite B have the same property:

PROPOSITION 4.7. *For every T-automaton ($\bigstar$) with finite B and $x \in X$, $[\![x]\!]_m : A^* \to B$ is regular.*

We will present the proof of this proposition after Corollary 8.2.

We are now ready to introduce fixpoint expressions for T-automata similar to (2.3).

Definition 4.8 (*Reactive expressions*). Let Σ be an algebraic signature and let B_0 be a finite set. *Reactive expressions* w.r.t. these data are closed terms δ defined according to the following grammar:

$$\begin{aligned} \delta &::= x \mid \gamma \mid f(\delta, \dots, \delta) && (x \in X, f \in \Sigma) \\ \gamma &::= \mu x.\, a_1.\delta \pitchfork \dots \pitchfork a_n.\delta \pitchfork \beta && (x \in X) \\ \beta &::= b \mid f(\beta, \dots, \beta) && (b \in B_0, f \in \Sigma) \end{aligned}$$

where we assume $A = \{a_1, \dots, a_n\}$ and an infinite collection of variables X. Free and bound variables here are defined in the standard way. We do not distinguish expressions equivalent under α-conversion (i.e. renaming of bound variables).

Notation 4.9.

(1) Let t be a Σ-term over $\{1, \dots, n\}$ (i.e. the numbers $1, \dots, n$ are identified as variables) and let $t_1, \dots, t_n$ be any Σ-terms. Then we write $t(t_1, \dots, t_n)$ for the substitution $t[t_1/1, \dots, t_n/n]$.

(2) For every Σ-algebra A (so, in particular for every $\mathbf{T}$-algebra, where Σ is part of a presentation of $\mathbf{T}$) we write $f^A : A^n \to A$ for the operation associated to $f : n \to 1$ from Σ. We also write $t^A : A^n \to A$ for the map evaluating the Σ-term t over $\{1, \dots, n\}$ in A.
(3) Finally, we shall sometimes call Σ-terms over a set X of variables simply Σ-terms.

Observe that a reactive expression can be uniquely represented in the form $t(e_1, \dots, e_n)$ where $e_1, \dots, e_n$ are reactive expressions starting with μ.

Let $\mathbf{T}$ be a finitary monad, generated by an algebraic theory $\mathcal{E}$ over the signature Σ and let B be a finitely generated $\mathbf{T}$-algebra over a finite set of generators B_0 (witnessed by the surjective $\mathbf{T}$-algebra morphism $h : TB_0 \to B$). Let us denote by E_{Σ,B_0} the set of all reactive expressions over Σ and B_0. We aim to define a reactive $\mathbf{T}$-algebra structure on a suitable quotient of E_{Σ,B_0}. First, notice that E_{Σ,B_0} is obviously a Σ-algebra. Then we introduce an L-transition structure on E_{Σ,B_0} as follows: notice that expressions b from the β-clause in Definition 4.8 are just Σ-terms on the generators from B_0. Recall also that B is a surjective image of TB_0 and let b^B be the image of $b \in B_0$ under

$$B_0 \xrightarrow{\eta_{B_0}} TB_0 \xrightarrow{\quad h \quad} B.$$

This extends to arbitrary Σ-terms over B_0 by putting $(t(b_1, \dots, b_k))^B = t^B(b_1^B, \dots, b_k^B)$. Then let us define

$$\begin{aligned} o(f(e_1, \cdots, e_n)) &= f^B(o(e_1), \dots, o(e_n)), \\ \partial_{a_i}(f(e_1, \cdots, e_n)) &= f(\partial_{a_i}(e_1), \dots, \partial_{a_i}(e_n)), \\ o(\mu x.\, (a_1.e_1 \pitchfork \dots \pitchfork a_n.e_n \pitchfork b)) &= b^B, \\ \partial_{a_i}(\mu x.\, (a_1.e_1 \pitchfork \dots \pitchfork a_n.e_n \pitchfork b)) &= e_i[\mu x.\, (a_1.e_1 \pitchfork \dots \pitchfork a_n.e_n \pitchfork b)/x]. \end{aligned} \tag{4.2}$$

This defines an L-transition structure $s : \mathsf{E}_{\Sigma,B_0} \to B \times (\mathsf{E}_{\Sigma,B_0})^A$ and so $\widehat{s} : \mathsf{E}_{\Sigma,B_0} \to B^{A^*}$ provides language semantics to expressions and a language equivalence relation $\sim$ on them according to Definition 2.1.

Notation 4.10. We overload notation and write $[\![e]\!] := \widehat{s}(e)$ (i.e. $[\![-]\!]$ with no subscripts) for the formal power series denoted by the expression e.

Note that the first two equations in (4.2) above imply that the L-transition structure s is a Σ-algebra homomorphism.

Remark 4.11. Recall that the category of Σ-algebras (and its full subcategory of all $\mathbf{T}$-algebras) has image factorizations. That means that every Σ-algebra morphism $f : A \to B$ can be factorized as a surjective Σ-algebra morphism $e : A \twoheadrightarrow C$ followed by an injective one $m : C \rightarrowtail B$. This factorization system has the usual *diagonalization* property: given a commutative square $m \cdot f = g \cdot e$ with m injective and e surjective we have a unique diagonal d with $m \cdot d = g$ and $d \cdot e = f$. See e.g. Adámek, Herrlich and Strecker [1990] for basics on factorization systems.

THEOREM 4.12. *The quotient $\mathsf{E}_{\Sigma,B_0}/{\sim}$ is a reactive $\mathbf{T}$-algebra whose L-coalgebra part is inherited from E_{Σ,B_0} and whose $\mathbf{T}$-algebra part is a quotient of the Σ-algebra structure on E_{Σ,B_0}.*

PROOF. Recall first that $\mathbf{T}$-algebras, being the variety of Σ-algebras satisfying the equations in $\mathcal{E}$, form a full subcategory of the category of Σ-algebras. We have seen that E_{Σ,B_0} is a coalgebra for the lifting of L to the category of Σ-algebras and that the final coalgebra for the lifting is B^{A^*} (its L-transition structure is a Σ-algebra morphism since it is a $\mathbf{T}$-algebra morphism). Thus, the language semantics map $[\![-]\!] : \mathsf{E}_{\Sigma,B_0} \to B^{A^*}$ is a Σ-algebra morphism. The quotient $\mathsf{E}_{\Sigma,B_0}/{\sim}$ is obtained by taking its factorization into a surjective followed by an injective Σ-algebra morphism:

$$\mathsf{E}_{\Sigma,B_0} \xrightarrow{\quad q \quad} \mathsf{E}_{\Sigma,B_0}/{\sim} \xrightarrow{\quad m \quad} B^{A^*}.$$

Since (the lifting of) L preserves monos we obtain an L-transition structure on the quotient by diagonalization:

$$\begin{array}{ccc} \mathsf{E}_{\Sigma,B_0} & \xrightarrow{\langle o,\partial\rangle} & L(\mathsf{E}_{\Sigma,B_0}) \\ {\scriptstyle q}\downarrow & & \downarrow{\scriptstyle Lq} \\ \mathsf{E}_{\Sigma,B_0}/{\sim} & \dashrightarrow & L(\mathsf{E}_{\Sigma,B_0}/{\sim}) \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle Lm} \\ B^{A^*} & \xrightarrow[\langle o,\partial\rangle]{} & L(B^{A^*}) \end{array}$$

More explicitly, the Σ-algebra structure on $\mathsf{E}_{\Sigma,B_0}/{\sim}$ is given for any operation $f : k \to 1$ in Σ by

$$f^{\mathsf{E}_{\Sigma,B_0}/{\sim}}\big([t_1]_\sim,\ldots,[t_k]_\sim\big) = \big[f(t_1,\ldots,t_k)\big]_\sim.$$

And the L-transition structure on $\mathsf{E}_{\Sigma,B_0}/{\sim}$ is given by

$$o([t]_\sim) = o(t) \quad \text{and} \quad \partial_a([t]_\sim) = [\partial_a(t)]_\sim.$$

Now since B^{A^*} is a $\mathbf{T}$-algebra and $\mathsf{E}_{\Sigma,B_0}/{\sim}$ is its sub-Σ-algebra, $\mathsf{E}_{\Sigma,B_0}/{\sim}$ is a sub-$\mathbf{T}$-algebra of B^{A^*} (since varieties are closed under subalgebras). Similarly, $L(\mathsf{E}_{\Sigma,B_0}/{\sim})$ is a sub-$\mathbf{T}$-algebra of $L(B^{A^*})$. It then follows that the L-transition structure on $\mathsf{E}_{\Sigma,B_0}/{\sim}$ is a $\mathbf{T}$-algebra morphism as a restriction of the L-transition structure on B^{A^*}. □

The following theorem is the main result of this section – it is a variant of the celebrated Kleene theorem for regular languages. Like its classical counterpart our theorem enables conversions from $\mathbf{T}$-automata to expressions and vice versa.

THEOREM 4.13 (KLEENE THEOREM). *For any reactive expression $e \in \mathsf{E}_{\Sigma,B_0}$ there is a corresponding $\mathbf{T}$-automaton (★) and a state $x \in X$ such that $[\![e]\!] = [\![x]\!]_m$. Conversely, for every $\mathbf{T}$-automaton (★) and state $x \in X$ there is an expression $e \in \mathsf{E}_{\Sigma,B_0}$ such that $[\![e]\!] = [\![x]\!]_m$.*

PROOF. ($\Rightarrow$) *From expressions to $\mathbf{T}$-automata.* Let $e \in \mathsf{E}_{\Sigma,B_0}$ and let us proceed with the definition of the corresponding $\mathbf{T}$-automaton. Recall that the grammar generating reactive expressions has γ- and δ-clauses and let us call a not necessarily closed expression a *γ-expression* if it matches the γ-clause.

We assume w.l.o.g. that distinct μ-operators bind distinct variables in e; this can be ensured by α-conversion. Let $X = \{x_1,\ldots,x_m\}$ be the set of variables occurring in e. For $i = 1,\ldots,m$, let

$$t_i = \mu x_i.\, a_1.t_1^i\theta_1^i \pitchfork \ldots \pitchfork a_n.t_n^i\theta_n^i \pitchfork b_i$$

be the uniquely determined subexpression of e with each t_j^i being Σ-terms (i.e. not containing μ) and each $t_j^i\theta_j^i$ being the maximal proper δ-subexpression of t_i (cf. Example 4.14 further below); consequently, the t_j^i are obtained from the maximal proper δ-subexpressions of e by replacing topmost occurrences of t_k (i.e. topmost subexpressions starting with μx_k) with x_k, and θ_j^i being the derived substitution sending every x_k introduced in this way to t_k. Note that the θ_j^i need not be total on X and note that the t_i, the θ_j^i and the t_j^i are uniquely determined by e. Without loss of generality we may assume that $t_1 = e$.

Starting with the triple

$$\{\,\},\ [\,],\ \{x_1 \doteq t_1\}, \tag{4.3}$$

where $\{\ \}$ denotes the emptyset and $[\ ]$ the empty substitution, we successively produce further triples of the form $I,\ \theta,\ S$, such that $I \subseteq \{x_1,\ldots,x_m\}$, θ is a substitution sending variables from

I to closed γ-expressions, and S is a set of formal equations of the form $x_i \doteq t_i$ such that all the free variable of each t_i are among I. A successor of such triple is (nondeterministically) produced by the rule

$$\frac{I,\ \theta,\ S \cup \{x_k \doteq t_k\}}{I \cup \{x_k\},\ \theta[t_k\theta/x_k],\ S \cup \{x_i \doteq t_i \mid \theta^k_j(x_i) = t_i\}} \quad (\{x_k \doteq t_k\} \notin S)$$

This procedure of successively applying the above rule eventually terminates with $S = \{\ \}$, for each step reduces the number of μ-operators that occur in the terms on the right-hand side of equations in S. Note that the above rule maintains the assumptions imposed on the triples $I,\ \theta,\ S$. Hence we obtain a triple $\{x_1, \ldots, x_m\},\ \rho,\ \{\ \}$ where the substitution ρ sends each x_i to a closed γ-expression, which we denote by e_i, i.e., $\rho = [e_1/x_1, \ldots, e_m/x_m]$, or equivalently $e_i = x_i\rho$ for $i = 1, \ldots, m$.

We assume henceforth the representation

$$e_i = \mu x_i.\, a_1.e^i_1 \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_n.e^i_n \mathbin{\pitchfork} b_i. \tag{4.4}$$

Observe that

$$e_i = t_i\rho, \tag{4.5}$$

which can be seen by induction as follows: if t_i was handled at the first iteration of the above procedure then

$$e_i = x_i\rho = x_i[t_i/x_i] = t_i = t_i\rho;$$

otherwise, by induction, we have

$$e_i = x_i\rho = x_i\theta[t_i\theta/x_i] = x_i[t_i/x_i]\theta = t_i\theta = t_i\rho,$$

where the middle equation holds by the properties of substitution.

Using the above definition of t_i, we obtain

$$e_i = \mu x_i.\, a_1.t^i_1\theta^i_1\rho_{-i} \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_n.t^i_n\theta^i_n\rho_{-i} \mathbin{\pitchfork} b_i$$

where ρ_{-i} agrees with ρ except that it leaves x_i unchanged. By comparing it with (4.4), we obtain for any i, j that $t^i_j\theta^i_j\rho_{-i} = e^i_j$ and therefore $e^i_j[e_i/x_i] = t^i_j\theta^i_j\rho$. Recall that for any $k = 1, \ldots, n$, θ^i_j sends x_k to t_k and ρ sends t_k to e_k, see (4.5). Therefore the composite substitution $\theta^i_j\rho$ sends each x_k to e_k, i.e., we have $\theta^i_j\rho = \rho$, whence

$$t^i_j\theta^i_j\rho = t^i_j\rho.$$

We have thus obtained

$$e^i_j[e_i/x_i] = t^i_j\rho = t^i_j[e_1/x_1, \ldots, e_m/x_m].$$

This allows us to restate the definitions for o and ∂ as follows:

$$\begin{aligned} o(t(e_1, \ldots, e_m)) &= t^B(b^B_1, \ldots, b^B_m), \\ \partial_{a_j}(t(e_1, \ldots, e_m)) &= t(t^1_j[e_1/x_1, \ldots, e_m/x_m], \ldots, t^m_j[e_1/x_1, \ldots, e_m/x_m]), \end{aligned}$$

for any Σ-term t over $\{1, \ldots, m\}$. Let $\rho_{a_j} = [t^1_j/x_1, \ldots, t^m_j/x_m]$ and inductively define $\rho_\epsilon = \mathsf{id}$, $\rho_{a_jw} = \rho_{a_j}\rho_w$. By induction we obtain

$$\begin{aligned} o(\partial_w(t(e_1, \ldots, e_m))) &= r^B(b^B_1, \ldots, b^B_m) \\ \text{where} \quad t(x_1\rho_w, \ldots, x_m\rho_w) &= r(x_1, \ldots, x_m). \end{aligned} \tag{4.6}$$

Suppose that $e = s(e_1, \ldots, e_m)$ with a Σ-term s and let $\widetilde{X} = \{x, x_1, \ldots, x_m\}$. We turn $T\widetilde{X}$ into a reactive $\mathbf{T}$-algebra. Recall that every element of $T\widetilde{X}$ can be written as $[t(x, x_1, \ldots, x_m)]_\equiv$, where t is a Σ-term and $[p]_\equiv$ denotes the equivalence class of the Σ-term p in $T\widetilde{X}$. Now let

$$o([t(x, x_1, \ldots, x_m)]_\equiv) = t^B(s^B(b^B_1, \ldots, b^B_m), b^B_1, \ldots, b^B_m),$$

$$\partial_{a_j}([t(x,x_1,\dots,x_m)]_{\equiv}) = \left[t(s(t_j^1,\dots,t_j^m),t_j^1,\dots,t_j^m)\right]_{\equiv}.$$

It is not difficult to see that the $\mathbf{T}$-algebra and the L-transition structures interact properly, i.e. o and the ∂_a are $\mathbf{T}$-algebra morphisms; in fact, $o = \alpha \cdot Tf$, where $\alpha : TB \to B$ is the $\mathbf{T}$-algebra on B and the map $f : \widetilde{X} \to B$ is defined by $f(x_i) = b_i$, $i = 1,\dots,m$, $f(x) = s^B(b_1^B,\dots,b_n^B)$; and $\partial_{a_j} = g_j^\star : T\widetilde{X} \to T\widetilde{X}$ where $g_j : \widetilde{X} \to T\widetilde{X}$ is the map defined by $g_j(x_i) = [t_j^i]_{\equiv}$ for $i = 1,\dots,m$ and $g_j(x) = [s(t_j^1,\dots,t_j^m)]_{\equiv}$. Note that the induced language semantics identifies x and $s(x_1,\dots,x_m)$, i.e. we have $[x]_{\equiv} \sim [s(x_1,\dots,x_n)]_{\equiv}$ (cf. Definition 2.1).

For a Σ-term t, by definition $\partial_{a_j}([t]_{\equiv}) = [t\rho_{a_j}]_{\equiv}$, so an easy induction shows that

$$o(\partial_w([t(x_1,\dots,x_m)]_{\equiv})) = r^B(b_1^B,\dots,b_m^B) \quad \text{where} \quad t(x_1\rho_w,\dots,x_n\rho_w) = r(x_1,\dots,x_m).$$

By comparing this to (4.6) we obtain by Proposition 2.2, that

$$[\![t(e_1,\dots,e_n)]\!] \sim [t(x_1,\dots,x_n)]_{\equiv}$$

Thus, specializing to $t = s$ we obtain

$$e = s(e_1,\dots,e_m) \sim [s(x_1,\dots,x_m)]_{\equiv} \sim [x]_{\equiv}.$$

By Remark 4.6(1), the constructed reactive $\mathbf{T}$-algebra is equivalent to a $\mathbf{T}$-automaton $\mathcal{m}$ for which we then clearly have $[\![e]\!] = [\![x]\!]_{\mathcal{m}}$.

($\Leftarrow$) *From $\mathbf{T}$-automata to expressions.* Suppose, we are given a $\mathbf{T}$-automaton ($\bigstar$). The generalized powerset construction yields a reactive $\mathbf{T}$-algebra over TX for which

$$o([t(x_1,\dots,x_m)]_{\equiv}) = t^B(b_1^B,\dots,b_m^B), \quad \partial_{a_j}([t(x_1,\dots,x_m)]_{\equiv}) = \left[t(t_j^1,\dots,t_j^m)\right]_{\equiv}, \tag{4.7}$$

where $b_i^B = o^{\mathcal{m}}(x_i) \in B$, t_j^i is a term representing $t^{\mathcal{m}}(a_j,x_i) \in TX$ and $[t]_{\equiv}$ denotes the equivalence class of the Σ-term t in TX. We successively build expressions $u_m,\dots,u_1$ such that all free variables of each u_i with $i > 1$ are in $\{x_1,\dots,x_{i-1}\}$ and u_1 is closed. Let

$$u_m = \mu x_m.\,(a_1.t_1^m \pitchfork \dots \pitchfork a_n.t_n^m \pitchfork b_m)$$
$$u_i = \mu x_i.\,(a_1.t_1^i[u_{i+1}/x_{i+1},\dots,u_m/x_m] \pitchfork \dots \pitchfork a_n.t_n^i[u_{i+1}/x_{i+1},\dots,u_m/x_m] \pitchfork b_i)$$

for all $i = m-1,\dots,1$, and let $e_1 = u_1$. We now apply the same construction to e_1 that we applied to e in the first part of the proof. Note that the Σ-terms t_j^i in the construction are precisely the t_j^i from (4.7) that we used to define the expressions u_i. Now the construction yields further expressions e_i, $i = 2,\dots,m$ and, for every i, expressions $e_1^i,\dots,e_n^i$ satisfying the identities

$$e_i = \mu x_i.\,(a_1.e_1^i \pitchfork \dots \pitchfork a_n.e_n^i \pitchfork b_i),$$
$$e_j^i[e_i/x_i] = t_j^i[e_1/x_1,\dots,e_n/x_n].$$

By the same argument as in the first part of the proof we obtain (4.6). Moreover, for the original reactive $\mathbf{T}$-algebra, also

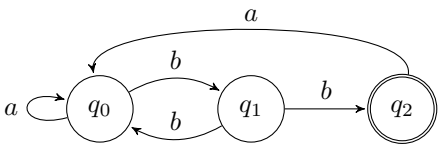
$$o(\partial_w([t(x_1,\dots,x_n)]_{\equiv})) = r^B(b_1^B,\dots,b_n^B) \quad \text{where} \quad t(x_1\rho_w,\dots,x_n\rho_w) = r(x_1,\dots,x_m),$$

and therefore we are done by Proposition 2.2. □

Example 4.14. Fig. 5 depicts a simple instance of the general correspondence established by Theorem 4.13 in the particular standard case of NFSM. For the expression for q_0, the subexpressions t_i, t_j^i and the substitutions θ_j^i are as follows (we omit empty substitutions θ_j^i):

$$t_1 = \mu x.\,(a.x \pitchfork b.\mu y.\,(a.\varnothing \pitchfork b.(x{+}\mu z.\,(a.x \pitchfork b.\varnothing \pitchfork \top)) \pitchfork \bot) \pitchfork \bot)$$
$$t_2 = \mu y.\,(a.\varnothing \pitchfork b.(x{+}\mu z.\,(a.x \pitchfork b.\varnothing \pitchfork \top)) \pitchfork \bot)$$
$$t_3 = \mu z.\,(a.x \pitchfork b.\varnothing \pitchfork \top)$$

$$\begin{cases} q_0 = a.q_0 \pitchfork b.q_1 \pitchfork \bot \\ q_1 = a.\varnothing \pitchfork b.(q_0 + q_2) \pitchfork \bot \\ q_2 = a.q_0 \pitchfork b.\varnothing \pitchfork \top \end{cases}$$



$$q_0 = \mu x.\,(a.x \pitchfork b.\mu y.\,(a.\varnothing \pitchfork b.(x + \mu z.\,(a.x \pitchfork b.\varnothing \pitchfork \top)) \pitchfork \bot) \pitchfork \bot)$$

Fig. 5. A $\mathcal{P}_\omega$-automaton over $A = \{a, b\}$, $B = \{\top, \bot\}$ as a system of recursive definitions (left); as a nondeterministic FSM (right); and as a reactive expression (bottom).

$$\begin{array}{lll} t_1^1 = x & t_2^1 = y & \theta_2^1 = [t_2/y] \\ t_1^2 = \emptyset & t_2^2 = x + z & \theta_2^2 = [t_3/z] \\ t_1^3 = x & t_2^3 = \emptyset & \end{array}$$

Furthermore, the triples obtained by successively applying the rule (4.3) are as follows:

I	ρ	S
$\{\ \}$	$[\,]$	$\{x \doteq t_1\}$
$\{x\}$	$[t_1/x]$	$\{y \doteq t_2\}$
$\{x, y\}$	$[t_1/x, t_2[t_1/x]/y]$	$\{z \doteq t_3\}$
$\{x, y, z\}$	$[t_1/x, t_2[t_1/x]/y, t_3[t_1/x, t_2[t_1/x]/y]/z]$	$\{\ \}$

5. T-AUTOMATA: EXAMPLES

As indicated in the previous section, a nondeterministic finite state machines (NFSM) is a specific case of a $\mathbf{T}$-automaton under $B = 2$ and $\mathbf{T} = \mathcal{P}_\omega$. More generally, we have the following definition.

Definition 5.1 (*Weighted* $\mathbf{T}$*-automata*). A *weighted* $\mathbf{T}$*-automaton* is a $\mathbf{T}$-automaton (★) with $\mathbf{T}$ being the semimodule monad for the semiring R (see Definition 3.13).

Let $\mathbf{T}$ be the semimodule monad for the semiring R. Besides the case $R = B = 2$, where we obtain NFSMs, we also obtain R-weighted automata [Droste et al. 2009] under $B = R$ (here B is the free $\mathbf{T}$-algebra finitely generated by $\{1\}$).

Weighted $\mathbf{T}$-automata can be further generalized as follows. We call a monad *additive* (cf. [Coumans and Jacobs 2013]) if the corresponding Σ-theory supports operations

$$+ : 2 \to 1 \qquad \text{and} \qquad \emptyset : 0 \to 1$$

subject to the axioms of commutative monoids. We call a $\mathbf{T}$-automaton *additive* if $\mathbf{T}$ is additive. Semimodule monads $\mathbf{T}_R$ are additive, of course. Besides the finite powerset monad $\mathbf{T} = \mathcal{P}_\omega$, which is the semimodule monad for the Boolean semiring $\{0, 1\}$, a simple example is the bag monad $\mathbf{T}_\mathbb{N}$, where $\mathbb{N}$ is the usual semiring of natural numbers. This monad assigns to every set X the finite multisets on X (i.e. the free commutative monoid on X).

Example 5.2 (*Probabilistic automata*). Rabin's probabilistic automata [Rabin 1963] can be modelled as weighted $\mathbf{T}$-automata over the semiring $[0, \infty)$ with the standard arithmetic operations.

In fact, a Rabin automaton is precisely a $\mathbf{T}$-automaton (★) with a fixed initial state x_0. Then given a *cut-point* $\lambda \in [0, 1)$, the set

$$\{w \in A^* \mid [\![x_0]\!]_m(w) > \lambda\}$$

is precisely the language accepted by the Rabin automaton with cut-point λ in the standard sense [Rabin 1963].

We now give one example of an additive $\mathbf{T}$-automaton, which is not a weighted $\mathbf{T}$-automaton.

Example 5.3. (Simple) Segala systems [Segala 1995; Segala and Lynch 1995] are systems combining probability and nondeterminism and are essentially coalgebras of transition type $\mathcal{P}(\mathcal{D}\times A) \cong (\mathcal{P}\mathcal{D})^A$ where $\mathcal{D}$ is the probability distribution functor. Unfortunately, $\mathcal{P}\mathcal{D}$ is not a monad [Dahlqvist and Neves 2017, Theorem 25]. However, the combination of probability and nondeterminism can be modelled by a monad $\mathbf{T}$ whose functorial part is the composition CM of two functors given as follows: for every X, MX consists of the finite valuations over X (cf. Definition 3.13); for any semimodule U, $C(U)$ consists of all subsets of U, which are nonempty and *convex*. Convexity of a set S here means that a convex combination $p_1\cdot\xi_1+\ldots+p_n\cdot\xi_n$, i.e. where $\sum_i p_i = 1$, belongs to S whenever $\xi_i \in S$ for every i. $\mathbf{T}$-automata for $\mathbf{T} = CM$ are automata with combined probabilistic and nondeterministic branching. Taking $B = CM1 = C[0,\infty)$, the set of all nonempty convex subsets of $[0,\infty)$, the semantics of a state of a $\mathbf{T}$-automaton is a formal power-series $A^* \to C[0,\infty)$. We leave the task of working out the relationship to Segala systems and their semantics for further work.

We will now show that additive $\mathbf{T}$-automata allow for a more relaxed syntax of reactive expressions. As before we fix a finite set $A = \{a_1,\ldots,a_n\}$ of actions.

Definition 5.4 (*Guardedness, Additive expressions*). Let Σ be the signature of the algebraic theory of the additive monad $\mathbf{T}$, and let B_0 be a finite set. We call an expression e defined by the grammar

$$\gamma ::= b \mid x \mid \mu x.\,\gamma \mid a.\gamma \mid f(\gamma,\ldots,\gamma) \qquad (a\in A, b\in B_0, f\in\Sigma) \tag{5.1}$$

guarded in x if one of the following inductive clauses apply:

— *(induction base)* $e \in B_0$, e is a variable distinct from x, $e = a.e'$, or $e = \mu x.\,e'$ for some expression e';
— *(induction step)* $e = f(e_1,\ldots,e_n)$ for some $e_1,\ldots,e_n$ guarded in x, or $e = \mu y.\,e'$ where $x \neq y$ and e' guarded in x.

An expression generated by (5.1) is an *open additive reactive expression* if for every of its subexpression $\mu x.\,e$, e is guarded in x. *Additive reactive expressions* are those open ones in which all variables are bound. We denote by A_{Σ,B_0} the set of additive reactive expression over Σ, B_0 and by $\mathsf{A}^O_{\Sigma,B_0}$ the corresponding set of open additive expressions.

PROPOSITION 5.5. *Let $\mathbf{T}$ be an additive monad and let B be a $\mathbf{T}$-algebra generated by the finite set B_0. Given a reactive expression we obtain an additive reactive expression by replacing recursively each $\Cap$ with $+$. Conversely, one can also transform any additive reactive expression to a reactive expression, and both transformation are mutually inverse modulo the semantic equivalence $\sim$.*

PROOF. (1) Let Σ be the signature of the Σ-theory of $\mathbf{T}$. First, we observe that A_{Σ,B_0} clearly carries a Σ-algebra structure. Moreover, it also carries an L-transition structure. In order to define it we first define an auxiliary normalization function n on (not necessarily closed) additive expressions as follows:

$$\mathsf{n}(f(e_1,\ldots,e_n)) = f(\mathsf{n}(e_1),\ldots,\mathsf{n}(e_n)) \qquad (f\neq +) \qquad \mathsf{n}(p+q) = p \qquad (\mathsf{n}(q)=\emptyset)$$
$$\mathsf{n}(p+q) = \mathsf{n}(p)+\mathsf{n}(q) \qquad (\mathsf{n}(p)\neq\emptyset,\ \mathsf{n}(q)\neq\emptyset) \qquad \mathsf{n}(p+q) = q \qquad (\mathsf{n}(p)=\emptyset)$$
$$\mathsf{n}(\mu x.e) = \mu x.\,\mathsf{n}(e) \qquad \mathsf{n}(a.e) = a.\mathsf{n}(e) \qquad \mathsf{n}(p) = p \qquad (p \text{ a variable or } p\in B_0)$$

Then we inductively define the L-transition structure on A_{Σ,B_0}:

$$o(b) = b^B \qquad o(\mu x.\,e) = o(e[\mu x.\,e/x]) \qquad o(a_i.e) = \emptyset^B$$
$$\partial_{a_i}(b) = \emptyset \qquad \partial_{a_i}(\mu x.\,e) = \partial_{a_i}(e[\mu x.\,e/x]) \qquad \partial_{a_i}(a_i.e) = \mathsf{n}(e),\ \partial_{a_i}(a_j.e) = \emptyset \qquad (i\neq j)$$
$$o(f(e_1,\ldots,e_n)) = f^B(o(e_1),\ldots,o(e_n)) \qquad \partial_{a_i}(f(e_1,\ldots,e_n)) = \mathsf{n}(f(\partial_{a_i}(e_1),\ldots,\partial_{a_i}(e_n)))$$

Our usage of n here is merely a technical trick to keep the proof elementary. Note that the clauses for $\mu x.\, e$ are well-founded due to guardedness.

We record the following simple properties of n:

$$\mathsf{n}(\mathsf{n}(p)) = \mathsf{n}(p) \tag{5.2}$$
$$\mathsf{n}(p+q) = \mathsf{n}(\mathsf{n}(p) + \mathsf{n}(q)) \tag{5.3}$$
$$\mathsf{n}(e[\mu x.\, t/y]) = \mathsf{n}(e)[\mathsf{n}(\mu x.\, t)/y] \tag{5.4}$$
$$\mathsf{n}(\partial_a(p)) = \partial_a(\mathsf{n}(p)) \tag{5.5}$$

where $p, q \in \mathsf{A}_{\Sigma,B_0}$. Identity (5.2) follows by structural induction over p. The only nontrivial case is $p = p_1 + p_2$ with $\mathsf{n}(p_1) \neq \emptyset$ and $\mathsf{n}(p_2) \neq \emptyset$ (note that in the third step below we use that, by induction, $\mathsf{n}(\mathsf{n}(p_i)) = \mathsf{n}(p_i) \neq \emptyset$):

$$\begin{aligned}
\mathsf{n}(\mathsf{n}(p)) &= \mathsf{n}(\mathsf{n}(p_1 + p_2)) \\
&= \mathsf{n}(\mathsf{n}(p_1) + \mathsf{n}(p_2)) && \text{// def. of } \mathsf{n} \\
&= \mathsf{n}(\mathsf{n}(p_1)) + \mathsf{n}(\mathsf{n}(p_2)) && \text{// def. of } \mathsf{n}, (5.2) \\
&= \mathsf{n}(p_1) + \mathsf{n}(p_2) && \text{// } (5.2) \\
&= \mathsf{n}(p_1 + p_2) && \text{// def. of } \mathsf{n} \\
&= \mathsf{n}(p).
\end{aligned}$$

Identity (5.3) then follows from (5.2) by case distinction: it is obvious if $\mathsf{n}(p) = \emptyset$ or $\mathsf{n}(q) = \emptyset$, otherwise $\mathsf{n}(p+q) = \mathsf{n}(\mathsf{n}(p+q)) = \mathsf{n}(\mathsf{n}(p)+\mathsf{n}(q))$. Identity (5.4) is a restricted form of substitution lemma, which can as usual be established by induction over the context e and the proof relies both on (5.2) and (5.3). Note, however that in our setting it does not hold more generally, e.g. with $e = b + y$, $\mathsf{n}(e[\emptyset/y]) = b \neq b + \emptyset = \mathsf{n}(e)[\mathsf{n}(\emptyset)/y]$. Finally, identity (5.5) follows from the previous identities by induction over p, in particular, the most difficult case $p = \mu x.\, e$ requires (5.4):

$$\begin{aligned}
\mathsf{n}(\partial_a(p)) &= \mathsf{n}(\partial_a(\mu x.\, e)) \\
&= \mathsf{n}(\partial_a(e[p/x])) && \text{// def. of } \partial_a \\
&= \partial_a(\mathsf{n}(e[p/x])) && \text{// ind. hypothesis} \\
&= \partial_a(\mathsf{n}(e)[\mathsf{n}(p)/x]) && \text{// } (5.4) \\
&= \partial_a(\mu x.\, \mathsf{n}(e)/x]) && \text{// def. of } \mathsf{n} \\
&= \partial_a(\mathsf{n}(p)). && \text{// def. of } \partial_a
\end{aligned}$$

Another case of interest in proving (5.5) is $p = p_1 + p_2$ under $\mathsf{n}(p_1) \neq \emptyset \neq \mathsf{n}(p_2)$:

$$\begin{aligned}
\partial_a(\mathsf{n}(p_1 + p_2)) &= \partial_a(\mathsf{n}(p_1) + \mathsf{n}(p_2)) && \text{// def. of } \mathsf{n} \\
&= \mathsf{n}(\partial_a(\mathsf{n}(p_1)) + \partial_a(\mathsf{n}(p_2))) && \text{// def. of } \partial_a \\
&= \mathsf{n}(\mathsf{n}(\partial_a(p_1)) + \mathsf{n}(\partial_a(p_2))) && \text{// ind. hypothesis} \\
&= \mathsf{n}(\partial_a(p_1) + \partial_a(p_2)) && \text{// } (5.3) \\
&= \mathsf{n}(\mathsf{n}(\partial_a(p_1) + \partial_a(p_2))) && \text{// } (5.2) \\
&= \mathsf{n}(\partial_a(p_1 + p_2)). && \text{// def. of } \partial_a
\end{aligned}$$

(2) By Definition 2.1, the above L-coalgebra structure on A_{Σ,B_0} induces a language semantics; again we write $[\![e]\!]$ for the formal power series denoted by $e \in \mathsf{A}_{\Sigma,B_0}$. Let us show that this semantics agrees with the semantics of E_{Σ,B_0}, that is $[\![e]\!] = [\![\mathsf{tr}(e)]\!]$ with $e \in \mathsf{E}_{\Sigma,B_0}$ and $\mathsf{tr} : \mathsf{E}_{\Sigma,B_0} \to \mathsf{A}_{\Sigma,B_0}$ defined inductively as follows:

$$\begin{aligned}
\mathsf{tr}(f(e_1, \ldots, e_n)) &= \mathsf{n}(f(\mathsf{tr}(e_1), \ldots, \mathsf{tr}(e_n))), & \mathsf{tr}(x) &= x, \\
\mathsf{tr}(\mu x.\, a_1.e_1 \pitchfork \ldots \pitchfork a_n.e_n \pitchfork s) &= \mu x.\, \mathsf{n}(a_1.\mathsf{tr}(e_1) + \ldots + a_n.\mathsf{tr}(e_n) + \mathsf{tr}(s)), & \mathsf{tr}(b) &= b.
\end{aligned}$$

Note that s in the bottom left equation is an arbitrary term in the theory of $\mathbf{T}$ according to the β-clause of the grammar in Definition 4.8. In fact, the above assigments define tr on expressions containing free variables and according to the γ and β-clauses of Definition 4.8. By case distinction it is straightforward to prove that for every (not necessarily closed) e we have

$$\mathsf{n}(\mathsf{tr}(e)) = \mathsf{tr}(e). \tag{5.6}$$

Moreover, we have the following property

$$\mathsf{tr}(e[t/x]) = \mathsf{n}(\mathsf{tr}(e)[\mathsf{tr}(t)/x]). \tag{5.7}$$

The proof of the latter is essentially straightforward but quite tedious. In order to show the desired equation $[\![e]\!] = [\![\mathsf{tr}(e)]\!]$, by Proposition 2.2, it suffices to check that tr is an L-coalgebra homomorphism, i.e.

$$\partial_{a_i}(\mathsf{tr}(e)) = \mathsf{tr}(\partial_{a_i}(e)) \quad \text{and} \quad o(\mathsf{tr}(e)) = o(e) \qquad (a_i \in A,\, e \in \mathsf{E}_{\Sigma,B_0})$$

This again follows by induction over the number of clauses recursively applied to define $o(e)$ and $\partial_{a_i}(e)$ and the proof relies on (5.2)–(5.5). E.g. for $e = f(e_1, \ldots, e_n)$ we calculate

$$\begin{aligned}
\partial_{a_i}(\mathsf{tr}(f(e_1,\ldots,e_n))) &= \partial_{a_i}(\mathsf{n}(f(\mathsf{tr}(e_1),\ldots,\mathsf{tr}(e_n)))) && \text{// def. of } \mathsf{tr}\\
&= \mathsf{n}(\partial_{a_i}(f(\mathsf{tr}(e_1),\ldots,\mathsf{tr}(e_n)))) && \text{// (5.5)}\\
&= \mathsf{n}(\mathsf{n}(f(\partial_{a_i}(\mathsf{tr}(e_1)),\ldots,\partial_{a_i}(\mathsf{tr}(e_n))))) && \text{// def. of } \partial_{a_i}\\
&= \mathsf{n}(f(\partial_{a_i}(\mathsf{tr}(e_1)),\ldots,\partial_{a_i}(\mathsf{tr}(e_n)))) && \text{// (5.2)}\\
&= \mathsf{n}(f(\mathsf{tr}(\partial_{a_i}(e_1)),\ldots,\mathsf{tr}(\partial_{a_i}(e_1)))) && \text{// ind. hypothesis}\\
&= \mathsf{tr}(f(\partial_{a_i}(e_1),\ldots,\partial_{a_i}(e_n))) && \text{// def. of } \mathsf{tr}\\
&= \mathsf{tr}(\partial_{a_i}(f(e_1,\ldots,e_n))), && \text{// def. (4.2) of } \partial_{a_i} \text{ on } \mathsf{E}_{\Sigma,B_0}
\end{aligned}$$

$$\begin{aligned}
o(\mathsf{tr}(f(e_1,\ldots,e_n))) &= o(\mathsf{n}(f(\mathsf{tr}(e_1),\ldots,\mathsf{tr}(e_n)))) && \text{// def. of } \mathsf{tr}\\
&= o(f(\mathsf{n}(\mathsf{tr}(e_1)),\ldots,\mathsf{n}(\mathsf{tr}(e_n)) && \text{// def. of } \mathsf{n}\\
&= o(f(\mathsf{tr}(e_1),\ldots,\mathsf{tr}(e_n))) && \text{// (5.6)}\\
&= f^B(o(\mathsf{tr}(e_1)),\ldots,o(\mathsf{tr}(e_n))) && \text{// def. of } o\\
&= f^B(o(e_1),\ldots,o(e_n)) && \text{// induction hypothesis}\\
&= o(f(e_1,\ldots,e_n)). && \text{// def. of } o
\end{aligned}$$

The remaining clauses do not cause any trouble and are handled in a similar fashion. For example, for $e = \mu x.\, a_1.e_1 \pitchfork \ldots \pitchfork a_n.e_n \pitchfork s$ we have by the definition of o

$$o(\mu x.\, a_1.e_1 \pitchfork \ldots \pitchfork a_n.e_n \pitchfork s) = s^B.$$

Starting at the right-hand side we have

$$\begin{aligned}
&o(\mathsf{tr}(\mu x.\, a_1.e_1 + \ldots + a_n.e_n + s)\\
&\quad = o(\mu x.\, \underbrace{\mathsf{n}(a_1.\mathsf{tr}(e_1) + \ldots + a_n.\mathsf{tr}(e_n) + \mathsf{tr}(s))}_{t}) && \text{// def. of } \mathsf{tr}\\
&\quad = o(\mathsf{n}(a_1.\mathsf{tr}(e_1) + \ldots + a_n.\mathsf{tr}(e_n) + \mathsf{tr}(s))[\mu x.\, t/x]). && \text{// def. of } o
\end{aligned}$$

If $\mathsf{n}(\mathsf{tr}(s)) = \emptyset$ then the latter evaluates to

$$o(a_1.\mathsf{n}(\mathsf{tr}(e_1))[\mu x.\, t/x] + \ldots + a_n.\mathsf{n}(\mathsf{tr}(e_n))[\mu x.\, t/x]) = \emptyset^B = s^B,$$

using the definition of n for the first equation, and the fact that $\mathsf{n}(\mathsf{tr}(s)) = \emptyset$ implies $s = \emptyset + \cdots + \emptyset$ for the second equation.

If $\mathsf{n}(\mathsf{tr}(s)) \neq \emptyset$ then, analogously,

$$\begin{aligned} & o(\mathsf{n}(a_1.\mathsf{tr}(e_1) + \ldots + a_n.\mathsf{tr}(e_n) + \mathsf{tr}(s))[\mu x.\, t/x]) \\ & \quad = o(a_1.\mathsf{n}(\mathsf{tr}(e_1))[\mu x.\, t/x] + \ldots + a_n.\mathsf{n}(\mathsf{tr}(e_n))[\mu x.\, t/x] + \mathsf{n}(\mathsf{tr}(s))) && \text{// def. of } \mathsf{n} \\ & \quad = (\mathsf{tr}(s))^B = s^B, \end{aligned}$$

where the last step is established by an easy induction (over terms s according to the β-clause in Definition 4.8).

Finally, we calculate:

$$\begin{aligned} & \mathsf{tr}(\partial_{a_i}(\mu x.\, a_1.e_1 \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_n.e_n \mathbin{\pitchfork} b)) \\ & \quad = \mathsf{tr}(e_i[\mu x.\, a_1.e_1 \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_n.e_n \mathbin{\pitchfork} b/x]) && \text{// def. of } \partial_{a_i} \\ & \quad = \mathsf{n}(\mathsf{tr}(e_i)[\mu x.\, \mathsf{n}(a_1.\mathsf{tr}(e_1) + \ldots + a_n.\mathsf{tr}(e_n) + b)/x]) && \text{// (5.7)} \\ & \quad = \mathsf{n}(\partial_{a_i}(\mu x.\, \mathsf{n}(a_1.\mathsf{tr}(e_1) + \ldots + a_n.\mathsf{tr}(e_n) + b))) && \text{// def. of } \partial_{a_i} \\ & \quad = \partial_{a_i}(\mathsf{n}(\mu x.\, \mathsf{n}(a_1.\mathsf{tr}(e_1) + \ldots + a_n.\mathsf{tr}(e_n) + b))) && \text{// (5.5)} \\ & \quad = \partial_{a_i}(\mathsf{tr}(\mu x.\, a_1.e_1 \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_n.e_n \mathbin{\pitchfork} b)). && \text{// def. of } \mathsf{tr} \end{aligned}$$

(3) In order to prove the desired converse in the statement of the proposition, we define a translation map $\overline{\mathsf{tr}}\colon \mathsf{A}_{\Sigma,B_0} \to \mathsf{E}_{\Sigma,B_0}$. In order to do this we first define an auxiliary map $\bar{o}$ on every expression according to (5.1) that is guarded in each of its variables; $\bar{o}$ works similarly as o but without interpreting $\emptyset$, f and b in B, whence delivering a term in the theory of $\mathbf{T}$ according to the β-clause of Definition 4.8:

$$\begin{array}{ll} \bar{o}(b) = b & \bar{o}(\mu x.\, e) = \bar{o}(e[\mu x\, e./x]) \\ \bar{o}(a.e) = \emptyset & \bar{o}(f(e_1, \ldots, e_n)) = f(\bar{o}(e_1), \ldots, \bar{o}(e_n)) \end{array}$$

Then $\bar{o}(e)$ is well-defined by guardedness of e. Similarly, we define auxiliary maps a^{-1} completely similarly as ∂_a; however, a^{-1} can be applied to expressions e containing free variables but which are still guarded in each of their variables. That means we do not (need to) define a^{-1} on variables x. Now we define $\overline{\mathsf{tr}}$ (on not necessarily closed expressions) as follows:

$$\begin{aligned} \overline{\mathsf{tr}}(x) &= x, \\ \overline{\mathsf{tr}}(b) &= \mu x.\, a_1.\emptyset \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_n.\emptyset \mathbin{\pitchfork} b, \\ \overline{\mathsf{tr}}(a_i.e) &= \mu x.\, a_1.\emptyset \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_i.\overline{\mathsf{tr}}(e) \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_n.\emptyset \mathbin{\pitchfork} \emptyset, \\ \overline{\mathsf{tr}}(f(e_1, \ldots, e_n)) &= f(\overline{\mathsf{tr}}(e_1), \ldots, \overline{\mathsf{tr}}(e_n)), \\ \overline{\mathsf{tr}}(\mu x.\, e) &= \mu x.\, a_1.\overline{\mathsf{tr}}(a_1^{-1}(e)) \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_n.\overline{\mathsf{tr}}(a_n^{-1}(e)) \mathbin{\pitchfork} \bar{o}(\mu x.\, e). \end{aligned}$$

Before we proceed we first need a substitution lemma similar to (5.7):

$$\overline{\mathsf{tr}}(e[t/x]) = \overline{\mathsf{tr}}(e)[\overline{\mathsf{tr}}(t)/x]. \tag{5.8}$$

We deduce $[\![e]\!] = [\![\overline{\mathsf{tr}}(e)]\!]$ for any $e \in \mathsf{A}_{\Sigma,B_0}$ from

$$o(\overline{\mathsf{tr}}(e)) = o(e) \quad \text{and} \quad \partial_a(\overline{\mathsf{tr}}(e)) = \overline{\mathsf{tr}}(\partial_a(e)) \ \text{ for every } a \in A.$$

We have, e.g. for $e = \mu x.\, t$,

$$\begin{aligned} \partial_{a_i}(\overline{\mathsf{tr}}(e)) &= \partial_{a_i}(\mu x.\, a_1.\overline{\mathsf{tr}}(a_1^{-1}(t)) \mathbin{\pitchfork} \ldots \mathbin{\pitchfork} a_n.\overline{\mathsf{tr}}(a_n^{-1}(t)) \mathbin{\pitchfork} o(e)) \\ &= \overline{\mathsf{tr}}(a_i^{-1}(t))[\overline{\mathsf{tr}}(e)/x] \\ &= \overline{\mathsf{tr}}(a_i^{-1}(t)[e/x]) && \text{// (5.8)} \\ &= \overline{\mathsf{tr}}(\partial_{a_i}(t[e/x])) && \text{// guardedness} \\ &= \overline{\mathsf{tr}}(\partial_{a_i}(e)). \end{aligned}$$

The remaining cases are verified routinely. □

Remark 5.6. We note that for weighted automata additive expressions can be equivalently converted to the familiar rational expressions from weighted automata theory. Suppose that B_0 is a one-element set, $\{1\}$ say, so $B = R$. Then we can define a composition operation $\bullet : \mathsf{A}^O_{\Sigma,B_0} \times \mathsf{A}^O_{\Sigma,B_0} \to \mathsf{A}^O_{\Sigma,B_0}$ inductively by:

$$x \bullet t = x \qquad 1 \bullet t = t \qquad (\mu x.\, e) \bullet t = \mu x.\, (e \bullet t)$$

$$(a.\, e) \bullet t = a.\, (e \bullet t) \qquad f(e_1, \ldots, e_n) \bullet t = f(e_1 \bullet t, \ldots, e_n \bullet t)$$

According to this definition we have $a.e = a.(1 \bullet e) = (a.1) \bullet e$, i.e. every expression $a.e$ can be expressed using $\bullet$ and expressions $a.1$ only. The signature Σ of the semimodule theory consists of one binary operation symbol $+$ and unary operation symbols, one for every $r \in R$, denoted $r \cdot -$. Thus, writing simply a for $a.1$ and r for $r \cdot 1$, an alternative syntax for additive reactive expressions can be defined by the following grammar:

$$\gamma ::= x \mid \mu x.\, \gamma \mid a \mid r \mid \gamma + \gamma \mid \gamma \bullet \gamma \qquad (a \in A, r \in R) \tag{5.9}$$

Guardedness becomes somewhat more complicated to formulate: t is guarded in x if x is contained in a subterm $t_l \bullet t_r$ of t in the right-hand subterm t_r, where the left-hand subterm t_l contains some letter $a \in A$. Again, we consider expressions in which in every subexpression $\mu x.\, e$, e is guarded in x and where all variables are bound. The syntax can be restricted further by requiring that in every expression $\mu x.\, t$, t is of the form $1 + e \bullet x$, where e is closed. Indeed, using the above sound equations, associativity of $\bullet$, and the following distributive laws

$$(s + t) \bullet e = s \bullet e + t \bullet e, \qquad e \bullet (s + u) = e \bullet s + e \bullet u,$$

the desired result can be shown by induction over the number of μ-operators as follows. Let $\mu x.\, t$ be an expression with t satisfying the induction hypothesis. Then t can be brought to the form $q + e \bullet x$ with q not containing x. Now we have

$$\begin{aligned} \mu x.\, t &= \mu x.\, (q + e \bullet x) \\ &= \mu x.\, (1 \bullet q + e \bullet (x \bullet q)) \\ &= \mu x.\, ((1 + e \bullet x) \bullet q) \\ &= (\mu x.\, (1 + e \bullet x)) \bullet q \end{aligned}$$

The usual notation for $\mu x.\, (1 + e \bullet x)$ is *Kleene star* e^*. Hence, by replacing $\mu x.\, \gamma$ with γ^* in the grammar (5.9) we thus arrive at the grammar of rational expressions used in the *Kleene-Schützenberger theorem* (see e.g. [Droste et al. 2009]).

5.1. Stack T-automata

Here and in later sections we turn our attention to a different kind of examples of $\mathbf{T}$-automata, where $\mathbf{T}$ is related to the store monad. A prominent instance are $\mathbf{T}$-automata where $\mathbf{T}$ is the stack monad (Definition 3.14), which model finite state machines manipulating a push-down store.

Definition 5.7 (*Stack* $\mathbf{T}$*-automaton*). A *stack* $\mathbf{T}$*-automaton* is a $\mathbf{T}$-automaton (★) for which

— $\mathbf{T}$ is the stack monad over Γ;
— B is the set of predicates over Γ^* consisting of all those $p \in 2^{\Gamma^*}$ for each of which there exists a k such that $p(wu) = p(w)$ whenever $|w| \geq k$;
— $\alpha^m : TB \to B$ is given by evaluation; it restricts the morphism

$$(2^{\Gamma^*} \times \Gamma^*)^{\Gamma^*} \xrightarrow{\mathsf{ev}^{\Gamma^*}} 2^{\Gamma^*},$$

where $\mathsf{ev} : 2^{\Gamma^*} \times \Gamma^* \to 2$ is the evaluation morphism:

$$\alpha^m(r, t)(s) = r(s)(t(s)).$$

Intuitively, $o^{\mathcal{m}} : X \to B \subseteq 2^{\Gamma^*}$ models the acceptance condition by final states and stack contents, that is, we can consider $w \in A^*$ to be jointly accepted by a stack $\mathbf{T}$-automaton $\mathcal{m}$, an initial state x_0, an initial stack symbol γ_0, a finite set of final states F and a set of final stack configurations S if $[\![x_0]\!]_{\mathcal{m}}(w)(\gamma_0) = \top$ where $o^{\mathcal{m}}(x)(s) = \top$ iff $x \in F$, $s \in S$. As B obeys constraints analogous to those of TX, scanning an unbounded portion of the stack by $o^{\mathcal{m}}$ is disallowed; the role of the algebraic structure $\alpha^{\mathcal{m}}$ is roughly to trace acceptance conditions backwards along the transition structure $t^{\mathcal{m}}$.

In terms of Σ-theories, B is finitely generated over the set of generators $B_0 = \{\bot, \top\}$ and as such is a quotient of $T2$ under additional laws: $push_i(\bot) = \bot$ and $push_i(\top) = \top$. The formal argument showing that B is indeed an algebra for the stack monad is as follows. By Corollary 3.12, the stack monad, being a submonad of the store monad over Γ^*, induces a submonad $\mathbf{P}$ of the reader monad over Γ^*. For this monad $\mathbf{P}$ we have that PX consists of those $r : \Gamma^* \to X$ for each of which there exists k such that for every $w \in \Gamma^*$ and $u \in \Gamma^*$, $r(wu) = r(w)$ whenever $|w| \geq k$. In particular, this makes $B = P2$ a $\mathbf{P}$-algebra and hence a $\mathbf{T}$-algebra.

The expected fact that stack $\mathbf{T}$-automata can be used as a replacement for deterministic push-down automata without silent transitions (viz *deterministic real-time push-down automata*) is justified by the following result.

THEOREM 5.8. *Let $\mathcal{m}$ be a stack $\mathbf{T}$-automaton. Given $x_0 \in X$ and $\gamma_0 \in \Gamma$,*

$$\{w \in A^* \mid [\![x_0]\!]_{\mathcal{m}}(w)(\gamma_0) = \top\} \tag{5.10}$$

is a real-time deterministic context-free language. Conversely, for any real-time deterministic context-free language $\mathcal{L} \subseteq A^*$ there exist a stack $\mathbf{T}$-automaton ($\bigstar$), an $x_0 \in X$, and a $\gamma_0 \in \Gamma$ such that $\mathcal{L}$ is the language in (5.10).

As we shall see in Theorem 6.7, one can obtain an analogous characterization of ordinary context-free languages (essentially because for nondeterministic push-down automata the restriction of being real-time is omissible).

For the proof of Theorem 5.8 we need an explicit description of the action of the language semantics map $[\![-]\!]_{\mathcal{m}}$ defined in Diagram (4.1) in terms of the given data of the $\mathbf{T}$-automaton $\mathcal{m}$.

LEMMA 5.9. *Given any $\mathbf{T}$-automaton ($\bigstar$), $x \in X$ and $w \in A^*$ then*

$$[\![x]\!]_{\mathcal{m}}(\epsilon) = o^{\mathcal{m}}(x), \qquad [\![x]\!]_{\mathcal{m}}(au) = \alpha^{\mathcal{m}}\,(\mathsf{do}\ y \leftarrow t^{\mathcal{m}}(a,x); \eta_X [\![y]\!]_{\mathcal{m}}(u))\,. \tag{5.11}$$

PROOF. Recall that the transition structure ι in (4.1) arises from $o : B^{A^*} \to B$ and $\partial_a : B^{A^*} \to B^{A^*}$ with $o(\sigma) = \sigma(\epsilon)$ and $\partial_a(\sigma) = \lambda w.\, \sigma(aw)$. Thus, we obtain the semantics map

$$[\![-]\!]_{\mathcal{m}} = \big(X \xrightarrow{\eta_X} TX \xrightarrow{\widehat{\mathcal{m}}^\sharp} B^{A^*}\big).$$

The commutativity of (4.1) can now equivalently be restated as the two equations

$$o([\![x]\!]_{\mathcal{m}}) = o^{\mathcal{m}}(x), \qquad \partial_a([\![x]\!]_{\mathcal{m}}) = \widehat{\mathcal{m}}^\sharp(t^{\mathcal{m}}(x,a)) \qquad \text{for every } x \in X \text{ and } a \in A.$$

The left equation implies the left of (5.11) since $o([\![x]\!]_{\mathcal{m}}) = [\![x]\!]_{\mathcal{m}}(\epsilon)$. For the second statement notice first that by the freeness of TX we have that $\widehat{\mathcal{m}}^\sharp$ is the unique $\mathbf{T}$-algebra morphism extending $[\![-]\!]_{\mathcal{m}}$. Thus, we have

$$\widehat{\mathcal{m}}^\sharp = \alpha \cdot T[\![-]\!]_{\mathcal{m}} : TX \to B^{A^*},$$

where α is the $\mathbf{T}$-algebra structure on B^{A^*}. Observe that $\alpha : T(B^{A^*}) \to B^{A^*}$ is given pointwise, i. e. α is the unique morphism satisfying

$$\mathsf{ev}_u \cdot \alpha = \big(T(B^{A^*}) \xrightarrow{T\mathsf{ev}_u} TB \xrightarrow{\alpha^{\mathcal{m}}} B\big),$$

for every $u \in A^*$, where $\mathsf{ev}_u : B^{A^*} \to B$ is the obvious evaluation at $u \in A^*$: $\mathsf{ev}_u(f) = f(u)$. It follows that for every word $u \in A^*$ we have

$$\widehat{m}^\sharp(-)(u) = \big(TX \xrightarrow{T(\mathsf{ev}_u \cdot [\![-]\!]_m)} TB \xrightarrow{\alpha^m} B\big);$$

indeed we have:

$$\widehat{m}^\sharp(-)(u) = \mathsf{ev}_u \cdot \widehat{m}^\sharp = \mathsf{ev}_u \cdot \alpha \cdot T[\![-]\!]_m = \alpha^m \cdot T\mathsf{ev}_u \cdot T[\![-]\!]_m = \alpha^m \cdot T(\mathsf{ev}_u \cdot [\![-]\!]_m)$$

and therefore

$$\begin{aligned} [\![x]\!]_m(au) &= \partial_a([\![x]\!]_m)(u) && \text{// definition of } \partial_a \\ &= \widehat{m}^\sharp(t^m(x,a))(u) && \text{// (4.1)} \\ &= (\alpha^m \cdot T(\mathsf{ev}_u \cdot [\![-]\!]_m))(t^m(x,a)). \end{aligned}$$

The last line is the desired right-hand side of the right equation in (5.11). □

Before we proceed with the proof of Theorem 5.8, let us recall that a deterministic pushdown automaton (dpda) M is determined by a transition function

$$\delta : Q \times (A + \{\epsilon\}) \times \Delta \to Q \times \Delta^* + \{\bot\}, \tag{5.12}$$

an initial stack symbol $\boxtimes \in \Delta$, an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$. Here Q is a finite set of all states, A is a finite alphabet of actions and Δ is a finite alphabet of stack symbols. The transition function δ is subject to the following restrictions: for every $x \in Q$, $\gamma \in \Delta$ (exclusively) either $\delta(x, \epsilon, \gamma) \neq \bot$ or $\delta(x, a, \gamma) \neq \bot$ for all $a \in A$. Automaton configurations and transitions over them are defined in the standard way.

A word w is recognized by M if there is a chain of transitions over automaton configurations that starts at $\langle x_0, \boxtimes\rangle$, consumes w, and finishes at some $\langle x_n, s_n\rangle$ with $x_n \in F$. A dpda M is called *real-time* if $\delta(x, \epsilon, \gamma) = \bot$ for every $x \in Q$, $\gamma \in \Delta$ and it is called *quasi-real-time* if there is n such that the following chain of transition is not admissible for any $x_1 \in Q$, $s_1 \in \Delta^*$ and $m > n$:

$$\langle x_1, s_1\rangle \xrightarrow{\epsilon} \langle x_2, s_2\rangle \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} \langle x_m, s_m\rangle$$

We will make use of the fact that the classes of languages recognized by real-time dpda and quasi-real-time dpda coincide [Harrison and Havel 1972].

PROOF OF THEOREM 5.8. Given (★) over a stack monad and a finite X, let us construct a quasi-real-time dpda M as follows. For any $x \in X$ and $a \in A$ let $n_{x,a}$ be the smallest $n \geq 1$ such that $t^m(x,a) : \Gamma^* \to X \times \Gamma^*$ sends any su with $s, u \in \Gamma^*$, $|s| = n$ to $\langle y, s'u\rangle$ where $\langle y, s'\rangle = t^m(x,a)(s)$. Analogously, let n_x be the smallest $n \geq 1$ such that $o^m(x) : \Gamma^* \to 2$ returns equal results on words agreeing on the first n letters. Note that the numbers $n_{x,a}$ and n_x exist by the definition of the stack monad. Let $m = \max\{n_x, \max_a n_{a,x}\}$. As the state space of M we take

$$Q = \big\{\langle x, s\boxtimes^k\rangle \mid x \in X, s \in \Gamma^*, |s| \leq m - k\big\}.$$

Let $\Delta = \Gamma + \{\boxtimes\}$. Then we define the transition function δ as follows:

(i) $\delta(\langle x, s\rangle, \epsilon, \gamma) = \langle\langle x, s\gamma\rangle, \epsilon\rangle$ if $\gamma \neq \boxtimes$ and $|s| < m$;
(ii) $\delta(\langle x, s\rangle, \epsilon, \boxtimes) = \langle\langle x, s\boxtimes\rangle, \boxtimes\rangle$ if $|s| < m$;
(iii) $\delta(\langle x, s\boxtimes^k\rangle, a, \gamma) = \langle\langle y, \epsilon\rangle, s'\gamma\rangle$ if $a \neq \epsilon$, $s \in \Gamma^{m-k}$ and $\langle y, s'\rangle = t^m(x,a)(s)$.

Finally, let

$$F = \{\langle x, s\boxtimes^k\rangle \in Q \mid o^m(x)(s) = 1, s \in \Gamma^{m-k}\}$$

be the set of accepting states of M. The intuitive motivation for the definition of M comes from the need to save portions of the stack as parts of the state. This is needed to model the behaviour of m, which unlike a standard pda can read several symbols from the stack at once and not just the top one.

For technical reasons it is convenient to assume that we always can transfer m symbols from the stack to the state. We ensure this by allowing the completion of the second component of the state with an appropriate number of symbols $\boxtimes$ added from the right if the stack happens to be shorter than m.

Our goal is to show that for any $w \in A^*$, $[\![x_0]\!]_m(w)(\gamma_0) = 1$ iff w is accepted by M with $\langle x_0, \gamma_0\rangle$ as the initial state. To that end we prove a (clearly) more general statement: for any $w \in A^*$, $x \in X$ and $s \in \Gamma^*$, $[\![x]\!]_m(w)(s) = 1$ iff there is a chain of transitions C over configurations of M corresponding to w, starting at $\langle\langle x, \epsilon\rangle, s\boxtimes\rangle$ and finishing in an accepting state. We proceed by induction over the length of w.

— Let $w = \epsilon$. Then by Lemma 5.9 $[\![x]\!]_m(w)(s) = o^m(x)(s) = o^m(x)(s')$ where s' is the prefix of length $\min\{|s|, n_x\}$ of s. Therefore, $[\![x]\!]_m(w)(s) = 1$ iff $\langle x, s'\boxtimes^k\rangle \in Q$ belongs to F with $k = m - |s'|$. On the other hand, by (i)–(ii), every chain C of transitions corresponding to $w = \epsilon$ and starting at $\langle\langle x, \epsilon\rangle, s\boxtimes\rangle$ must be a prefix of the following chain:

$$\langle\langle x, \epsilon\rangle, s\boxtimes\rangle \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} \langle\langle x, s'\boxtimes^k\rangle, u\boxtimes\rangle$$

where $s = s'u$ and $k = m - |s'|$. Clearly, C leads to an accepting configuration iff $\langle x, s'\boxtimes^k\rangle$ is an accepting state.

— Let $w = au$. Then by Lemma 5.9,

$$\begin{aligned}[\![x]\!]_m(w)(s) &= \alpha^m\left(\mathsf{do}\ y \leftarrow t^m(x,a); \eta_X [\![y]\!]_m(u)\right)(s)\\ &= [\![y]\!]_m(u)(s') \qquad \text{where } \langle y, s'\rangle = t^m(x,a)(s).\end{aligned}$$

The latter is equal to 1 iff $[\![y]\!]_m(u)(s') = 1$ where $\langle y, s'\rangle = t^m(x,a)(s)$. By the induction hypothesis $[\![y]\!]_m(u)(s') = 1$ iff there is a chain of transitions C corresponding to u, starting at $\langle\langle y, \epsilon\rangle, s'\boxtimes\rangle$ and finishing in an accepting state. We shall show that there is a chain of transitions C' starting in $\langle\langle x, \epsilon\rangle, s\boxtimes\rangle$ and finishing in an accepting state. There are two cases: (1) if $|s| < m$ then we obtain C' by prepending C with

$$\langle\langle x, \epsilon\rangle, s\boxtimes\rangle \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} \langle\langle x, s\boxtimes^k\rangle, \boxtimes\rangle \xrightarrow{a} \langle\langle y, \epsilon\rangle, s'\boxtimes\rangle,$$

where $k = m - |s|$; (2) if $|s| \geq m$ let $s = s''w$ with $|s''| = m$ and let $t^m(x,a)(s'') = (\hat{y}, \hat{s})$. Then since $t^m(x,a)(s''u) = (\hat{y}, \hat{s}u)$ holds by the properties of $t^m(x,a) : \Gamma^* \to X \times \Gamma^*$, we know that $\hat{y} = y$ and $\hat{s}u = s'$. So we obtain C' by prepending C with

$$\langle\langle x, \epsilon\rangle, s\boxtimes\rangle \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} \langle\langle x, s''\rangle, u\boxtimes\rangle \xrightarrow{a} \langle\langle \hat{y}, \epsilon\rangle, \hat{s}u\boxtimes\rangle = \langle\langle y, \epsilon\rangle, s'\boxtimes\rangle.$$

Conversely, given a chain of transitions C' for w from $\langle\langle x, \epsilon\rangle, s\boxtimes\rangle$ and leading to a final state, then it must be a chain C starting at $\langle\langle y, \epsilon\rangle, s'\boxtimes\rangle$ prepended by one of the above two prefixes (depending on $|s|$). This completes the induction and the proof of the first part of the theorem.

In order to show the second part of the claim, suppose we are given a real-time deterministic pda M with a transition function (5.12), an initial state $q_0 \in Q$, a set of accepting states $F \subseteq Q$ and an initial stack symbol $\boxtimes$. Let us define a T-automaton ($\bigstar$) with $X = Q + \{\bot\}$ and T being the stack monad over Δ as follows: for every $q \in X$, $s \in \Delta^*$, $a \in A$, $o^m(q)(s) = 1$ iff $q \in F$ and

$$\begin{aligned}t^m(q,a)(\epsilon) &= t^m(\bot,a)(\gamma s) = \langle \bot, \epsilon\rangle\\ t^m(q,a)(\gamma s) &= \langle q', s's\rangle \qquad \text{where } \langle q', s'\rangle = \delta(q,a,\gamma).\end{aligned}$$

Let us show by induction over the length of $w \in A^*$ that for every $q \in Q$, $s \in \Delta^*$ an accepting configuration is reachable from $\langle q, s\rangle$ by w iff $[\![q]\!]_m(w)(s) = 1$.

— Let $w = \epsilon$. Then $\langle q, s\rangle$ is accepting iff $q \in F$ iff $o^m(q)(s) = 1$. By Lemma 5.9, the latter is equivalent to $[\![q]\!]_m(w)(s) = 1$.

— Let $w = au$. Then an accepting configuration is reachable from $\langle q, s\rangle$ iff $\langle q, s\rangle \xrightarrow{a} \langle q', s'\rangle$ for some $\langle q', s'\rangle$ from which an accepting configuration is reachable by u. By induction hypothesis

and by definition of t^m, an equivalent formulation is as follows: $[\![q']\!]_m(u)(s') = 1$ where $\langle q', s'\rangle = t^m(q,a)(s)$. On the other hand, by Lemma 5.9,

$$\begin{aligned}[\![q]\!]_m(w)(s) &= \alpha^m\left(\mathsf{do}\ q' \leftarrow t^m(q,a); \eta_X[\![q']\!]_m(u)\right)(s)\\ &= [\![q']\!]_m(u)(s') \qquad \text{where } \langle q', s'\rangle = t^m(q,a)(s),\end{aligned}$$

i.e. also $[\![q]\!]_m(w)(s) = 1$ iff $[\![q']\!]_m(u)(s') = 1$ where $\langle q', s'\rangle = t^m(q,a)(s)$.

As a result, the language recognized by M is equal to (5.10) under $x_0 = q_0$ and $\gamma_0 = \boxtimes$. □

6. MONAD TENSORS FOR COMBINING STORE AND NONDETERMINISM

Tensor products of monads (resp. algebraic theories) have been introduced by Freyd [1966] in the context of universal algebra. Later, computational relevance of this operation has been demonstrated by Hyland et al. [2007]. Here, we use tensors of monads as a tool for studying $\mathbf{T}$-automata, where $\mathbf{T}$ combines (several kinds of) store with nondeterminism.

Definition 6.1 (*Tensor*). Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two algebraic theories. Then the tensor product $\mathcal{E} = \mathcal{E}_1 \otimes \mathcal{E}_2$ is the algebraic theory, whose equations are obtained by joining the equations of $\mathcal{E}_1$ and $\mathcal{E}_2$ and adding for every $f : n \to 1$ of $\mathcal{E}_1$ and every $g : m \to 1$ of $\mathcal{E}_2$ the following axiom

$$f(g(x_1^1,\dots,x_m^1),\dots,g(x_1^n,\dots,x_m^n)) = g(f(x_1^1,\dots,x_1^n),\dots,f(x_m^1,\dots,x_m^n))$$

called the *tensor laws*. Given two finitary monads $\mathbf{T}_1$ and $\mathbf{T}_2$, their tensor product $\mathbf{T}_1 \otimes \mathbf{T}_2$ arises from the algebraic theory $\mathcal{E}_{\mathbf{T}_1} \otimes \mathcal{E}_{\mathbf{T}_2}$. Note that the embedding of terms and equations of $\mathcal{E}_i$, $i = 1, 2$, into $\mathcal{E}_1 \otimes \mathcal{E}_2$ gives rise to monad morphisms $\mathbf{T}_i \to \mathbf{T}_1 \otimes \mathbf{T}_2$ called *tensor injections*.

Intuitively, the tensor product of two monads captures a noninterfering combination of the corresponding computational effects. In the present work we shall use two kinds of tensor products: (1) tensors with *submonads of the store monad* (see Example 3.10) and (2) tensors with *semimodule monads* (see Definition 3.13). This allows us to combine nondeterminism with one or several stores in one monad.

It has been shown in [Hyland et al. 2007] that tensoring with the store monad is equivalent to the application of the store monad transformer sending any monad $\mathbf{T}$ to the *store monad transform* $\mathbf{T}_S$ whose functorial part is given by $T_S X = T(X \times S)^S$. Here we establish a similar result about the stack monad (Defnition 3.14).

PROPOSITION 6.2. *Let $\mathbf{S}$ be the stack monad over Γ. Then for any finitary monad $\mathbf{T}$, $\mathbf{S} \otimes \mathbf{T}$ is the submonad $\mathbf{R}$ of the store monad transform of $\mathbf{T}$ with Γ^* as the store, identified by the following condition: $p : \Gamma^* \to T(X \times \Gamma^*)$ is in RX if*

$$\exists k \in \mathbb{N}.\, \forall s \in \Gamma^k.\, \forall u \in \Gamma^*.\, p(su) = \mathsf{do}\,\langle x, s'\rangle \leftarrow p(s); \eta_{X\times\Gamma^*}\langle x, s'u\rangle. \tag{6.1}$$

PROOF. Note that (6.1) is equivalent to

$$\exists k \in \mathbb{N}.\, \forall s, u \in \Gamma^*.\, |s| \geq k \Rightarrow p(su) = \mathsf{do}\,\langle x, s'\rangle \leftarrow p(s); \eta_{X\times\Gamma^*}\langle x, s'u\rangle. \tag{6.2}$$

Indeed, the implication (6.2) ⇒ (6.1) is obvious. For the converse one, let k be as in (6.1), let $s, u \in \Gamma^*$ and let $|s| \geq k$. Then $s = s'w$ for suitable $s' \in \Gamma^k$, $w \in \Gamma^*$, and

$$\begin{aligned}p(su) &= p(s'wu)\\ &= \mathsf{do}\,\langle x, s''\rangle \leftarrow p(s'); \eta_{X\times\Gamma^*}\langle x, s''wu\rangle\\ &= \mathsf{do}\,\langle x, s''\rangle \leftarrow (\mathsf{do}\,\langle x, s''\rangle \leftarrow p(s'); \eta_{X\times\Gamma^*}\langle x, s''w\rangle); \eta_{X\times\Gamma^*}\langle x, s''u\rangle\\ &= \mathsf{do}\,\langle x, s''\rangle \leftarrow (\mathsf{do}\,\langle x, s''\rangle \leftarrow p(s'w); \eta_{X\times\Gamma^*}\langle x, s''\rangle); \eta_{X\times\Gamma^*}\langle x, s''u\rangle\\ &= \mathsf{do}\,\langle x, s''\rangle \leftarrow p(s); \eta_{X\times\Gamma^*}\langle x, s''u\rangle.\end{aligned}$$

We next check that (6.1) does indeed identify a submonad of the aforementioned store monad transform. First, for any $x \in X$, $p = \eta_X(x)$ satisfies (6.1) with $k = 0$. Then, for every

$f : X \to (T(Y \times \Gamma^*))^{\Gamma^*}$, such that for every $x \in X$, $f(x)$ satisfies (6.1) with some k_x, and for every $p : \Gamma^* \to T(X \times \Gamma^*)$, satisfying (6.1) with some k, we must show that $f^\star(p)$ also satisfies (6.1). Note that for $s \in \Gamma^k$,

$$\begin{aligned} f^\star(p)(su) &= \mathsf{do}\,\langle x, s'\rangle \leftarrow p(su); f(x)(s') \\ &= \mathsf{do}\,\langle x, s'\rangle \leftarrow (\mathsf{do}\,\langle x, s'\rangle \leftarrow p(s); \eta_{X\times\Gamma^*}\langle x, s'u\rangle); f(x)(s') \\ &= \mathsf{do}\,\langle x, s'\rangle \leftarrow p(s); f(x)(s'u). \end{aligned}$$

Since by assumption, $\mathbf{T}$ is finitary, for some finite $X' \subseteq X$ and $m \in \mathbb{N}$, $p(s) \in T(X' \times \Gamma^*)$. By (6.2), for $\hat{k} = \max\{k_x \mid x \in X'\}$, and $u \in \Gamma^{\hat{k}}$, we continue as follows:

$$\begin{aligned} f^\star(p)(suw) &= \mathsf{do}\,\langle x, s'\rangle \leftarrow p(s); f(x)(s'uw) \\ &= \mathsf{do}\,\langle x, s'\rangle \leftarrow p(s); \langle y, s''\rangle \leftarrow f(x)(s'u); \eta_{Y\times\Gamma^*}\langle y, s''w\rangle \\ &= \mathsf{do}\,\langle y, s''\rangle \leftarrow (\mathsf{do}\,\langle x, s'\rangle \leftarrow p(s); f(x)(s'u)); \eta_{Y\times\Gamma^*}\langle y, s''w\rangle \\ &= \mathsf{do}\,\langle y, s'\rangle \leftarrow f^\star(p)(su); \eta_{Y\times\Gamma^*}\langle y, s'w\rangle. \end{aligned}$$

That is, we have proven (6.1) for $f^\star(p)$ with $k + \hat{k}$.

Let us refer to the stack theory over $\Gamma = \{\gamma_1, \dots, \gamma_n\}$ as $\mathcal{E}$ and to the theory corresponding to the monad $\mathbf{T}$ as $\mathcal{T}$. We define a semantics of the theory $\mathcal{E} \otimes \mathcal{T}$ over $\mathbf{R}$ as follows:

$$\begin{aligned} (\!|pop|\!)_{\mathbf{R}}(\epsilon) &= \eta_{N\times\Gamma^*}\langle n+1, \epsilon\rangle, & (\!|pop|\!)_{\mathbf{R}}(\gamma_i w) &= \eta_{N\times\Gamma^*}\langle i, w\rangle, \\ (\!|push_i|\!)_{\mathbf{R}}(w) &= \eta_{1\times\Gamma^*}\langle 1, \gamma_i w\rangle, & (\!|f|\!)_{\mathbf{R}}(w) &= \mathsf{do}\ j \leftarrow (\!|f|\!)_{\mathbf{T}}; \eta_{M\times\Gamma^*}\langle j, w\rangle, \end{aligned}$$

where η denoted the unit of the monad $\mathbf{T}$, $N = \{1, \dots, n+1\}$, 1 in the subscript of η denotes the set $\{1\}$, i ranges from 1 to n, f ranges over the operations of $\mathcal{T}$, and $M = \{1, \dots, m\}$, where m is the arity of f. We proceed by verifying the properties prescribed by Theorem 3.9. This verification is analogous to the proof of Theorem 3.16 and hence we discuss only the specific features of the case at hand.

— *Soundness.* We have to verify soundness of (i) the stack theory, (ii) the equations from $\mathcal{T}$, and (iii) the tensor laws. Soundness of (i) is verified exactly as in Theorem 3.16. Soundness of (ii) immediately follows from soundness of $\mathcal{T}$ over $\mathbf{T}$. Finally, soundness of (iii) is verified directly for the $push$ and for pop operations. For $push_i$ we have for every set X

$$\begin{aligned} &(\!|push_i(f(x_1, \dots, x_k))|\!)_{RX}(w) \\ &\quad= (\mathsf{do}\,(\!|push_i|\!)_{\mathbf{R}}; j \leftarrow (\!|f|\!)_{\mathbf{R}}; (\!|x_j|\!)_{RX})(w) \\ &\quad= \mathsf{do}\,\langle x, u\rangle \leftarrow \eta_{1\times\Gamma^*}\langle 1, \gamma_i w\rangle; \langle j, v\rangle \leftarrow (\!|f|\!)_{\mathbf{R}}(u); (\!|x_j|\!)_{RX}(v) \\ &\quad= \mathsf{do}\,\langle j, v\rangle \leftarrow (\!|f|\!)_{\mathbf{R}}(\gamma_i w); (\!|x_j|\!)_{RX}(v) \\ &\quad= \mathsf{do}\ j \leftarrow (\!|f|\!)_{\mathbf{T}}; \langle x, v\rangle \leftarrow \eta_{M\times\Gamma^*}\langle j, \gamma_i w\rangle; (\!|x_j|\!)_{RX}(v) \\ &\quad= \mathsf{do}\ j \leftarrow (\!|f|\!)_{\mathbf{T}}; (\!|x_j|\!)_{RX}(\gamma_i w) \end{aligned}$$

and

$$\begin{aligned} &(\!|f(push_i(x_1), \dots, push_i(x_k))|\!)_{RX}(w) \\ &\quad= (\mathsf{do}\ j \leftarrow (\!|f|\!)_{\mathbf{R}}; (\!|push_i|\!)_{\mathbf{R}}; (\!|x_j|\!)_{RX})(w) \\ &\quad= \mathsf{do}\,\langle j, v\rangle \leftarrow (\!|f|\!)_{\mathbf{R}}(w); \langle x, u\rangle \leftarrow (\!|push_i|\!)_{\mathbf{R}}(v); (\!|x_j|\!)_{RX}(u) \\ &\quad= \mathsf{do}\ j \leftarrow (\!|f|\!)_{\mathbf{T}}; (y, v) \leftarrow \eta_{M\times\Gamma^*}\langle j, w\rangle; \langle x, u\rangle \leftarrow (\!|push_i|\!)_{\mathbf{R}}(v); (\!|x_j|\!)_{RX}(u) \\ &\quad= \mathsf{do}\ j \leftarrow (\!|f|\!)_{\mathbf{T}}; \langle x, u\rangle \leftarrow (\!|push_i|\!)_{\mathbf{R}}(w); (\!|x_j|\!)_{RX}(u) \\ &\quad= \mathsf{do}\ j \leftarrow (\!|f|\!)_{\mathbf{T}}; \langle x, u\rangle \leftarrow \eta_{1\times\Gamma^*}\langle 1, \gamma_i w\rangle; (\!|x_j|\!)_{RX}(u) \\ &\quad= \mathsf{do}\ j \leftarrow (\!|f|\!)_{\mathbf{T}}; (\!|x_j|\!)_{RX}(\gamma_i w) \end{aligned}$$

We leave the verification for pop to the reader.

— *Expressiveness.* Let $p : \Gamma^* \to T(X \times \Gamma^*)$ be an element of RX under some parameter k. We construct a term p_k over X such that $(\!|p_k|\!) = p$ by induction over k adapting the construction from Theorem 3.16.

— Let $k = 0$ and note that $p(\epsilon) \in T(X \times \Gamma^*)$. Let q be a term over $X \times \Gamma^*$ for which $(\!|q|\!)_{T(X\times\Gamma^*)} = p(\epsilon)$ and let p_0 be obtained from q by replacing any $\langle x, \gamma_{i_m} \dots \gamma_{i_1}\rangle \in X \times \Gamma^*$ by the term $push_{i_1}(\cdots(push_{i_m}(x))\cdots)$.

— If $k > 0$ then we build p_k from p_{k-1} in the same way as in Theorem 3.16.

— *Completeness.* Suppose we are given s and t such that $(\!|s|\!)_{RX} = (\!|t|\!)_{RX}$. Let us normalize both s and t using the equations of the stack theory oriented from left to right and additionally the rules:

$$push_i(f(x_1,\dots,x_m)) \to f(push_i(x_1),\dots,push_i(x_m)) \tag{6.3}$$

$$pop(x_1,\dots,x_n,f(y_1,\dots,pop(z_1,\dots,z_n,z),\dots,y_m)) \to pop(x_1,\dots,x_n,f(y_1,\dots,z,\dots,y_m)) \tag{6.4}$$

where $f(1,\dots,m)$ is an m-ary term in the signature of $\mathcal{T}$. Note that the obtained system is strongly normalizing because every rule either decreases the height of the term, or keeps it the same, but propagates the $push$ operator downwards. Except for the last rule, by definition, the respective equations belong to $\mathcal{E} \otimes \mathcal{T}$. The equation corresponding to the last rule also belongs to $\mathcal{E} \otimes \mathcal{T}$, for by **(pop-push)**, **(pop-pop)**, and by tensor laws:

$$\begin{aligned}
&pop(x_1,\dots,x_n,f(y_1,\dots,pop(z_1,\dots,z_n,z),\dots,y_m))\\
&\quad= pop(x_1,\dots,x_n,f(pop(push_1(y_1),\dots,push_n(y_1),y_1),\dots,\\
&\qquad\qquad pop(z_1,\dots,z_n,z),\dots,\\
&\qquad\qquad pop(push_1(y_m),\dots,push_n(y_m),y_m)))\\
&\quad= pop(x_1,\dots,x_n,pop(f(push_1(y_1),\dots,z_1,\dots,push_1(y_m)),\dots,\\
&\qquad\qquad f(push_1(y_1),\dots,z_n,\dots,push_1(y_m)),f(y_1,\dots,z,\dots,y_m)))\\
&\quad= pop(x_1,\dots,x_n,f(y_1,\dots,z,\dots,y_m)).
\end{aligned}$$

It suffices to prove that $s = t \in \mathcal{E} \otimes \mathcal{T}$ for normal s and t, and this is done by induction over the total number of operations distinct from $push$ occurring in s and t. Let f and g be terms (possibly a single variable) in the signature of $\mathcal{T}$ such that $s = f(s_1,\dots,s_m)$ and $t = g(t_1,\dots,t_l)$ and where each of the terms $s_1,\dots,s_m$ and $t_1,\dots,t_l$ is either a variable or has an operation of the stack theory at the top.

— If none of the terms $s_1,\dots,s_m,t_1,\dots,t_l$ contains the pop operation on top, then by normality each of these terms must be an application of a sequence of the $push$ operations to a variable. Hence, by the definition of our semantics we obtain

$$\begin{aligned}
(\!|s|\!)_{RX}(\epsilon) &= (\!|f(\langle x_1,w_1\rangle,\dots,\langle x_m,w_m\rangle)|\!)_{T(X\times\Gamma^*)},\\
(\!|t|\!)_{RX}(\epsilon) &= (\!|g(\langle y_1,u_1\rangle,\dots,\langle y_l,u_l\rangle)|\!)_{T(X\times\Gamma^*)},
\end{aligned}$$

where $\langle x_i, w_i\rangle = (\!|s_i|\!)_{RX}(\epsilon)$ and $\langle y_j, u_j\rangle = (\!|t_j|\!)_{RX}(\epsilon)$. It follows that

$$f(\langle x_1,w_1\rangle,\dots,\langle x_m,w_m\rangle) = g(\langle y_1,u_1\rangle,\dots,\langle y_l,u_l\rangle)$$

is provable in $\mathcal{T}$, and the desired proof of $s = t \in \mathcal{E} \otimes \mathcal{T}$ can be obtained from that proof by substituting every $\langle x_i, w_i\rangle$ by s_i and every $\langle y_j, u_j\rangle$ by t_j.

— Otherwise, suppose that $s_j = pop(\dots, s')$ for some $j \in \{1,\dots,m\}$. Using the laws of **R** we have that

$$\begin{aligned}
s &= f(s_1,\dots,s_j,\dots,s_m)\\
&= f(s_1,\dots,pop(\dots,s'),\dots,s_m)
\end{aligned}$$

$$
\begin{aligned}
&= f(pop(push_1(s_1),\dots,push_n(s_1),s_1),\dots,pop(\dots,s'),\dots,\\
&\quad pop(push_1(s_m),\dots,push_n(s_m),s_m)) && \text{// \textbf{(pop-push)}}\\
&= pop(f(push_1(s_1),\dots,push_1(s_m)),\dots,\\
&\quad f(push_n(s_1),\dots,push_n(s_m)),f(s_1,\dots,s',\dots,s_m)). && \text{// tensor law}
\end{aligned}
$$

Now substitute the last term for the right-hand s in

$$s = pop(push_1(s),\dots,push_n(s),s)$$

and use **(pop-pop)** and **(pop-push)** to conclude

$$
\begin{aligned}
s &= pop(push_1(s),\dots,push_n(s),f(s_1,\dots,s',\dots,s_m)),\\
t &= pop(push_1(t),\dots,push_n(t),t).
\end{aligned}
\tag{6.5}
$$

By (possibly repeated) application of the rule (6.4), we may replace $f(s_1,\dots,s',\dots,s_m)$ and t in the right-hand arguments by terms $\tilde{s}$ and $\tilde{t}$, respectively, that do not contain pop. Thus we obtain

$$
\begin{aligned}
s &= pop(push_1(s),\dots,push_n(s),\tilde{s}\\
t &= pop(push_1(t),\dots,push_n(t),\tilde{t}),
\end{aligned}
$$

whence by soundness and since $(\!|s|\!)_{RX} = (\!|t|\!)_{RX}$ we have

$$(\!|pop(push_1(s),\dots,push_n(s),\tilde{s})|\!)_{RX} = (\!|pop(push_1(t),\dots,push_n(t),\tilde{t})|\!)_{RX}.$$

Therefore, by Lemma 3.15 (which is easily seen to be valid over $\mathbf{R}$), we obtain

$$(\!|push_1(s)|\!)_{RX} = (\!|push_1(t)|\!)_{RX},\dots,(\!|push_n(s)|\!)_{RX} = (\!|push_n(t)|\!)_{RX}$$

and

$$(\!|\tilde{s}|\!)_{RX} = (\!|\tilde{t}|\!)_{RX}.$$

Note that each $push_i(s)$ can be renormalized, and since s_j has the pop operation on top, by **(push-pop)** the total number of operations distinct from $push$ decreases at least by one. Hence, using the induction hypothesis, we obtain $push_i(s) = push_i(t) \in \mathcal{E} \otimes \mathcal{T}$ for every i. Analogously, $f(s_1,\dots,s',\dots,s_m)$ has one pop operator less that s. Moreover, rewriting the former and t, respectively, in their contexts in (6.5) by the rule (6.4) might only reduce the number of pop operators further, while the number of all other operators remains unchanged. Therefore, we obtain $\tilde{s} = \tilde{t} \in \mathcal{E} \otimes \mathcal{T}$, and by standard equational reasoning we have

$$
\begin{aligned}
s &= pop(push_1(s),\dots,push_n(s),\tilde{s})\\
&= pop(push_1(t),\dots,push_n(t),\tilde{t})\\
&= t,
\end{aligned}
$$

i.e., we obtain that $s = t \in \mathcal{E} \otimes \mathcal{T}$ as desired. □

Using Proposition 6.2, one can combine two stacks by computing the tensor square of the stack monad. The resulting monad $\mathbf{T}$ is a submonad of the store monad for $S = \Gamma^* \times \Gamma^*$, whence elements of TX are certain maps of the form

$$\langle r, t_1, t_2\rangle : \Gamma^* \times \Gamma^* \to X \times \Gamma^* \times \Gamma^*.$$

This allows one to define $\mathbf{T}$-stack automata over two and more stacks analogously to the one-stack case from Definition 5.7. Before we do this formally in Definition 6.5 we briefly discuss forming tensors with semimodule monads.

PROPOSITION 6.3 ([FREYD 1966]). *The tensor product of any finitary monad with a semimodule monad is again a semimodule monad.*

Remark 6.4. Proposition 6.3, in conjunction with Proposition 6.2, offers two perspectives on machines with memory and nondeterminism. On the one hand, we shall, for example, consider the tensor product of $\mathcal{P}_\omega$ with the stack monad to model (nondeterministic) push-down automata. As Proposition 6.2 indicates, this monad embeds into the monad with functorial part $TX = \mathcal{P}_\omega(X \times \Gamma^*)^{\Gamma^*}$. On the other hand, by Proposition 6.3, this tensor product is equivalent to a semimodule monad. A rough intuition about this change of perspective can be gained from the isomorphism $\mathcal{P}(X \times \Gamma^*)^{\Gamma^*} \cong \mathcal{P}(\Gamma^* \times \Gamma^*)^X$ relating "nondeterministic" stateful computations over X and values over X weighted in the semiring $\mathcal{P}(\Gamma^* \times \Gamma^*)$.

Definition 6.5 (*Multi-stack nondeterministic* $\mathbf{T}$*-automaton*). A *multi-stack nondeterministic* $\mathbf{T}$*-automaton* is a $\mathbf{T}$-automaton ($\bigstar$) for which

— $\mathbf{T}$ is the tensor of m copies of the stack monad with $\mathcal{P}_\omega$;
— B is the set of m-ary predicates over Γ^* consisting of all those $p \in 2^{\Gamma^* \times \cdots \times \Gamma^*}$ for each of which there is a k such that for every $w_i \in \Gamma^k$ and $u_i \in \Gamma^*$, $i = 1, \ldots, m$, we have $p(w_1 u_1, \ldots, w_m u_m) = p(w_1, \ldots, w_m)$;
— for every $s \in (\Gamma^*)^m$, $f : (\Gamma^*)^m \to \mathcal{P}_\omega(B \times (\Gamma^*)^m) \in TB$

$$\alpha^{\mathbf{m}}(f)(s) = 1 \quad \text{iff} \quad \exists s' \in (\Gamma^*)^m.\ \exists p \in B.\ (p, s') \in f(s) \wedge p(s').$$

To see that B in Definition 6.5 is indeed a $\mathbf{T}$-algebra, let us deduce the following corollary of Lemma 3.11.

COROLLARY 6.6. *Let* $\mathbf{T}_S$ *be the nondeterministic store monad over* S *(i. e.* $TX = \mathcal{P}_\omega(X \times S)^S$*) and let* $\mathbf{R}_S$ *be the nondeterministic reader monad over* S *(i.e.* $R_S X = \mathcal{P}_\omega(X)^S$*). For every submonad* $\mathbf{T}$ *of* $\mathbf{T}_S$*, the monad morphism* α *sending any* $f : S \to \mathcal{P}_\omega(X \times S)$ *to* $\mathcal{P}_\omega(\pi_1) \cdot f : S \to \mathcal{P}_\omega(X)$ *restricts to a submonad of* $\mathbf{R}_S$.

Recall that by Proposition 6.2, the tensor of $\mathcal{P}_\omega$ with m copies of the stack monad over Γ^* is the submonad $\mathbf{T}$ of the nondeterministic store monad over $(\Gamma^*)^m$ identified by the following condition: $f : (\Gamma^*)^m \to \mathcal{P}_\omega(X \times (\Gamma^*)^m) \in TX$ iff there exists a k such that whenever $|u_1| \geq k, \ldots, |u_m| \geq k$ then

$$f(u_1 w_1, \ldots, u_m w_m) = \{\langle x, u_1' w_1, \ldots, u_m' w_m \rangle \mid \langle x, u_1', \ldots, u_m' \rangle \in f(u_1, \ldots, u_m)\}.$$

This induces a submonad $\mathbf{R}$ of the nondeterministic reader monad over $(\Gamma^*)^m$ identified by the following condition: $f : (\Gamma^*)^m \to \mathcal{P}_\omega(X) \in TX$ iff there exists a k such that whenever $|u_1| \geq k, \ldots, |u_m| \geq k$ then $f(u_1 w_1, \ldots, u_m w_m) = f(u_1, \ldots, u_m)$. The $\mathbf{T}$-algebra used in Definition 6.5 is thus obtained by taking $X = 1$.

We are now ready to prove the following result.

THEOREM 6.7. *For any* m *let* $\mathcal{L}_m$ *be the following class of all languages*

$$\{w \in A^* \mid [\![x_0]\!]_{\mathbf{m}}(w)(\gamma_0, \ldots, \gamma_0) = \top\} \tag{6.6}$$

with $\mathbf{m}$ *ranging over nondeterministic multistack* $\mathbf{T}$*-automata with* m *stacks,* x_0 *ranging over the state space of* $\mathbf{m}$ *and* γ_0 *ranging over* Γ. *Then*

(1) $\mathcal{L}_1$ *is the class of context-free languages;*
(2) for all $m > 2$, $\mathcal{L}_m$ *is the class of nondeterministic linear time languages* $\mathsf{NTIME}(n)$*;*
(3) $\mathcal{L}_2$ *sits properly between* $\mathcal{L}_1$ *and* $\mathcal{L}_3$.

PROOF. The proof is completely analogous to the proof of Theorem 5.8. We outline the main distinctions. In lieu of quasi-real-time deterministic pda we use nondeterministic push-down quasi-real-time (NPDQRT) machines (see [Book and Greibach 1970]). The transition function δ of such a machine has type

$$\delta : Q \times (A + \{\epsilon\}) \times \Delta^m \to \mathcal{P}_\omega(Q \times (\Delta^*)^m). \tag{6.7}$$

This function is subject to the condition of being *quasi-real-time*, i.e. there is a global bound on the lengths of ϵ-transition chains over machine configurations.

Two acceptance conditions for NPDQRT are possible: (i) by final states and (ii) by the empty stack. It is a standard exercise to make sure that a language accepted by empty storage can be accepted by final states. In fact, the construction for ordinary PDAs (see e.g. [Hopcroft et al. 2001]) also works for NPDQRT: for a given PDA P one forms a PDA P' with a fresh initial stack symbol γ_0' and a new inital state that pushes the original initial stack symbol on all stacks and then proceeds to the initial state of P. In addition, P' has one final state that can be reached from all states by an (internal) ϵ-transition if the stack content is $(\gamma_0', \dots, \gamma_0')$ (which corresponds to configurations of P with all stacks empty). Clearly, this construction preserves quasi real-timeness.

Conversely, for every m, (i) can be modelled by (ii), i.e. a language accepted by final states can be accepted by the empty stack: for $m = 1$, we obtain standard push-down automata for which the equivalence of (i) and (ii) is well-known [Rozenberg and Salomaa 1997]; for $m = 2$ this is shown in [Ginsburg and Harrison 1968]; for any $m > 2$, by [Book and Greibach 1970], the languages recognized under (ii) are exactly $\mathsf{NTIME}(n)$ and since for quasi-real-time machines the depths of all stacks is linearly bounded, these can be purged in linear time once a final state is reached.

As shown in [Li 1985], the class of languages recognized by NPDQRT with $m = 2$ is properly between context-free and $\mathsf{NTIME}(n)$.

It remains to show that for every m the languages recognized by nondeterministic multistack $\mathbf{T}$-automata with m stacks are the same as the languages recognized by NPDQRT with m stacks with the acceptance condition chosen at pleasure.

As in Theorem 5.8, given a nondeterministic multistack $\mathbf{T}$-automaton $\mathcal{M}$ with m stacks we identify a global bound n for the depths of the stack prefixes accessed at one step and then model $\mathcal{M}$ by an NPDQRT M over the state space

$$Q = \big\{\langle x, s_1 \boxtimes^{k_1}, \dots, s_m \boxtimes^{k_m}\rangle \mid x \in X, s_i \in \Gamma^*, |s_i| \leq n - k_i\big\}.$$

The stack alphabet Δ is $\Gamma + \{\boxtimes\}$, the transition function is given as in Theorem 5.8 by changing the number of elements in tuples Q and by allowing for nondeterminism. The acceptance condition is chosen to be by the following final states:

$$F = \{\langle x, s_1 \boxtimes^{k_1}, \dots, s_m \boxtimes^{k_m}\rangle \in Q \mid o^{\mathcal{M}}(x)(s) = 1, s_i \in \Gamma^{n-k_i}\}.$$

It then follows along the same lines as in the proof of Theorem 5.8 that for every $w \in A^*$, $(\!|x_0|\!)_{\mathcal{M}}(w)(\gamma_0, \dots, \gamma_0) = 1$ iff w is accepted by M with $\langle x_0, \gamma_0, \dots, \gamma_0\rangle$ as the initial configuration.

In order to show the second part of the claim, assume that M is a NPDQRT with m stacks, a transition function (6.7), an initial state $q_0 \in Q$, a set of accepting states $F \subseteq Q$ and an initial stack symbol $\boxtimes$. According to [Book and Greibach 1970], we assume that M is *real-time*, that means that no internal transitions are present.

We define a nondeterministic $\mathbf{T}$-automaton over m stacks with $X = Q$ and with stack symbols Δ as follows: for any $q \in X$, $s_i \in \Delta^*$, $a \in A$, $o^{\mathcal{M}}(q)(s_1, \dots, s_m) = 1$ iff $q \in F$ and

$$\begin{aligned} t^{\mathcal{M}}(q,a)(s_1, \dots, s_m) &= \emptyset && \text{(if } s_i = \epsilon \text{ for some } i\text{)} \\ t^{\mathcal{M}}(q,a)(\gamma_1 s_1, \dots, \gamma_m s_m) &= \{\langle q', s_1' s_1, \dots, s_m' s_m\rangle \mid \\ &\qquad \langle q', s_1', \dots, s_m'\rangle \in \delta(q, a, \gamma_1, \dots \gamma_m)\} && \text{(otherwise)} \end{aligned}$$

A similar argument as in Theorem 5.8 then shows that for every $w \in A^*$, $q \in Q$ and $s \in \Delta^*$ an accepting configuration is reachable from $\langle q, s_1, \dots, s_m\rangle$ by w iff $[\![q]\!]_{\mathcal{M}}(w)(s_1, \dots, s_m) = 1$. □

Theorem 6.7 shows, on the one hand, that the coalgebraic formalization of nondeterministic push-down automata as nondeterministic $\mathbf{T}$-automata over one stack is adequate in the sense that it recognizes the same class of languages. On the other hand, it indicates the boundaries of the present model: it seems unlikely to capture languages beyond $\mathsf{NTIME}(n)$ (e.g. all recursive ones) by a

computationally feasible class of $\mathbf{T}$-automata. This is not surprising in view of the early work on (quasi-)real-time recognizable languages [Book and Greibach 1970], which underlies the proof of Theorem 6.7. We return to this issue in Section 8 where we provide an extension of the present semantics that allows us to capture language classes up to recursively enumerable ones.

We conclude this section with a corollary of Theorem 6.7 and Proposition 2.2. It is well known that equivalence of context-free languages is undecidable; in fact, it is Π^0_1-complete (the non-halting problem for arbitrary Turing machine can be encoded as an equality of certain context-free languages [Hartmanis 1967]). We will use this to prove a similar completeness result for the equivalence of reactive expressions. We say that a Σ-algebra B over a set of generators B_0 is *effectively presented (over Σ and B_0)* if Σ and B_0 are recursive sets and the set

$$\{(t,s) \mid t,s \text{ are closed terms over } \Sigma, B_0 \text{ with } t^B = s^B\}$$

is decidable (recall Notation 4.9(2)). The language equivalence problem for reactive expressions is then the following decision problem: given recursive sets Σ and B_0, an effectively presented $\mathbf{T}$-algebra B, and two reactive expressions e_1 and e_2 in E_{Σ,B_0}, decide if $e_1 \sim e_2$ (cf. (4.2)).

COROLLARY 6.8. *The language equivalence of reactive expressions is Π^0_1-complete.*

PROOF. The fact that language equivalence of reactive expressions is in Π^0_1, i.e. co-r.e., follows from Proposition 2.2: if two reactive expressions e and u are not language equivalent, we can eventually detect this by finding a suitable word $w \in A^*$ for which $o(\partial_w(e)) \neq o(\partial_w(u))$. Here we rely on our effectiveness assumption, for, by the definitions (4.2), both $o(\partial_w(e))$ and $o(\partial_w(u))$ are terms over Σ and B_0 evaluated over B.

To prove Π^0_1-hardness, let us show how to reduce the equivalence problem of context-free languages to the current equivalence problem of reactive expressions. Given two context-free languages L_1 and L_2 recognized by two pushdown automata over a stack alphabet $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$, we provide an instance of our problem with $B \subseteq 2^{\Gamma^*}$ being the $\mathbf{T}$-algebra from Definition 6.5 for the nondeterministic stack theory $\mathbf{T}$ over one stack. We need to prove that B is effectively presented. First note that both Σ and B_0 are finite, specifically, B_0 is the two-element set $\{\top, \bot\}$. Indeed, Σ consists of the operation symbols pop, $push_i$, $i = 1, \ldots, n$, $+$ and $\emptyset$, and we recall from Definition 6.5 that B consists of precisely those predicates p over Γ^* for each of which there is a k such that for every $w \in \Gamma^k$ and $u \in \Gamma^*$, $p(wu) = p(w)$. Hence, each predicate p in B can be finitely represented, e.g. by the list of words w in Γ^k with $p(w) = \top$. In order to check that $t^B = s^B$ for a given pair of terms t, s over Σ, B_0 we first compute the two predicates t^B, s^B and then verify that they are equal. Indeed, for the latter we only need to verify $(t^B)(w) = (s^B)(w)$ for finitely many $w \in \Gamma^*$, which is a decidable problem, and for the former we need to verify that the algebra operations on B are computable. Using the definition of α^m in Definition 6.5 and the interpretation of the nondeterministic stack theory in the monad $\mathbf{T}$, which is a submonad of the store monad transform $\mathcal{P}_\omega(X \times \Gamma^*)^{\Gamma^*}$ (cf. the proof of Proposition 6.2), we verify that for every p, q, p_i, $i = 1, \ldots n$, in B we have

$$\begin{aligned} pop^B(p_1, \ldots, p_n, q)(\epsilon) &= q(\epsilon) \\ pop^B(p_1, \ldots, p_n, q)(\gamma_i w) &= p_i(w) \\ push_i^B(p)(w) &= p(\gamma_i w) \\ (p +^B q)(w) &= p(w) \vee q(w) \\ \emptyset^B(w) &= \bot. \end{aligned}$$

Hence, the above operations are clearly computable as desired.

By Theorem 6.7(2), we have two nondeterministic stack $\mathbf{T}$-automata m_1 and m_2 and states x_1, x_2, respectively, in them such that

$$L_i = \{w \in A^* \mid [\![x_i]\!]_{m_i}(w)(\gamma_1) = \top\}, \quad \text{for } i = 1, 2.$$

Further, by Theorem 4.13, we obtain reactive expressions e_i in E_{Σ,B_0} such that $[\![e_i]\!] = [\![x_i]\!]_{m_i}$ for $i = 1, 2$. Thus, we have $L_1 = L_2$ iff $\lambda w.\, [\![e_1]\!](w)(\gamma_1) = \lambda w.\, [\![e_2]\!](w)(\gamma_1)$. Of course, the latter is not equivalent to $\lambda w.\, [\![e_1]\!](w)(s) = \lambda w.\, [\![e_2]\!](w)(s)$ for *all* $s \in \Gamma^*$. However, it is easy to modify e_1 and e_2 to obtain this property: let for $i = 1, 2$,

$$e_i' = pop(pop(\emptyset, \ldots, \emptyset, push_1(e_i)), \emptyset, \ldots, \emptyset).$$

Then, clearly, $[\![e_1']\!] = [\![e_2']\!]$ iff $\lambda w.\, [\![e_1]\!](w)(\gamma_1) = \lambda w.\, [\![e_2]\!](w)(\gamma_1)$. □

One can also consider the language equivalence problem of reactive expressions for a fixed Σ and B; a very similar argument than the one in the previous proof then shows that the language equivalence of reactive expressions for the algebra $B \subseteq 2^{\Gamma^*}$ of Definition 6.5 is Π_1^0-complete.

However, for other Σ and B (coming from a type of $\mathbf{T}$-automaton), the language equivalence of reactive expressions is decidable, e.g. for finite Σ and B this follows from Theorem 4.13 and Proposition 4.7, for the identity monad $\mathbf{T}$ and any recursive set B (in this case $\mathbf{T}$-automata are simply Moore automata with output in B), for the finite powerset monad $\mathbf{T} = \mathcal{P}_\omega$ and $B = 2$ (for which $\mathbf{T}$-automata are classical nondeterministic automata), or for the monad $\mathbf{T}$ assigning to a set the set of formal linear combinations with coefficients from a fixed field k and $B = k$ (for which $\mathbf{T}$-automata are weighted automata over k). Identifying further monads $\mathbf{T}$ and algebras B for which the language equivalence for reactive $\mathbf{T}$-expressions is decidable is an interesting question for future work.

7. CONTEXT-FREE LANGUAGES AND VALENCE AUTOMATA

Throughout this section we assume that R is a semiring finitely generated by the set R_0; B is an R-semimodule finitely generated by the set B_0; and $\mathbf{T}_R$ is the semimodule monad for R.

By Proposition 6.3, a nondeterministic $\mathbf{T}$-automaton over one stack is a specific case of a weighted $\mathbf{T}$-automaton (Definition 5.1). In this form it is rather similar to *valence automata*, another example of a machine previously studied in the literature (e.g. [Render and Kambites 2009; Kambites 2009]). We present the corresponding algebraic theories side by side and explain how valence automata can be formalised as $\mathbf{T}$-automata.

Example 7.1 (*Nondeterministic stack theory*). The *nondeterministic stack theory* is obtained by tensoring $\mathcal{E}_{\mathcal{P}_\omega}$, i.e. the theory of commutative, idempotent monoids, with the stack theory. The result is a semimodule theory over an idempotent semiring R presented by the generators o_i, u_i, $i = 1, \ldots, |\Gamma|$ and e and the following relations:

$$u_i o_i = 1 \quad u_i o_j = 0 \quad u_i e = 0 \quad o_1 u_1 + \ldots + o_n u_n + e = 1 \quad e o_i = 0 \quad ee = e \quad (i \neq j)$$

The corresponding unary operations of the semimodule theory are denoted by $pop_i = \overline{o_i} : 1 \to 1$, $push_i = \overline{u_i} : 1 \to 1$ and $empty = \overline{e} : 1 \to 1$ (cf. the notation of Definition 3.13). It is straightforward to relate the nondeterministic stack theory and the presented semimodule theory by giving two translations defining the operations of one theory in terms of operations of the other. First the unary operations pop_i and $empty$ of the semimodule theory determine pop:

$$pop(x_1, \ldots, x_n, y) = pop_1(x_1) + \ldots + pop_n(x_n) + empty(y).$$

Conversely, pop_i and $empty$ can be defined from pop:

$$pop_i(x) = pop(\emptyset, \ldots, x, \ldots, \emptyset, \emptyset) \qquad empty(x) = pop(\emptyset, \ldots, \emptyset, x)$$

(x is on the i-the position in the sequence on the left). It is then straightforward to prove that the axioms of the nondeterministic stack theory and semimodule theory for R, respectively, are satisfied for the operations as defined by the translations.

Example 7.2 (*Nondeterministic monoid action theory*). The *nondeterministic monoid action theory* is obtained by tensoring the theory $\mathcal{E}_{\mathcal{P}_\omega}$ with the theory of M-actions of the monoid $(M, \cdot, 1)$ (see Example 3.10). As shown in [Hyland et al. 2007], the corresponding monad $\mathbf{T}$ maps a set X to

$\mathcal{P}_\omega(M \times X)$ and has the unit $\eta_X : x \mapsto \{(1, x)\}$ and the Kleisli lifting given by extending a map $f : X \to \mathcal{P}_\omega(M \times Y)$ to $f^\star : \mathcal{P}_\omega(M \times X) \to \mathcal{P}_\omega(M \times Y)$ with

$$f^\star(S) = \{\, (m \cdot n, y) \mid \exists x.\, (m, x) \in S \wedge (n, y) \in f(x) \,\}.$$

Note that the theory corresponding to $\mathbf{T}$ is the semimodule theory for the semiring $R = \mathcal{P}_\omega(M)$ with addition given by $\cup$ with unit $\emptyset$ and multiplication given by $S \cdot S' = \{m \cdot n \mid m \in S, n \in S'\}$ for any finite subsets S and S' of M with unit $\{1\}$.

Now let the monoid $(M, \cdot, 1)$ be fixed. The idea of valence automata over M is to use the monoid structure to model various kinds of stores (stack(s), counter(s), etc.). Recall e.g. from [Render and Kambites 2009; Kambites 2009] that a valence automaton over M is a tuple $\mathcal{A} = (X, M, A, \delta, q_0, F)$ where X is a finite set of states, δ is finite subset of $X \times A^* \times M \times X$ of transitions, $q_0 \in X$ is an initial state and $F \subseteq X$ a set of final states. This induces a transition relation $\Rightarrow$ on $X \times A^* \times M$ as usual by defining $(p, w, m) \Rightarrow (q, wu, mn)$ if there exists $(p, u, n, q) \in \delta$. The language accepted by a given valence automaton $\mathcal{A}$ is

$$L(\mathcal{A}) = \{\, w \in A^* \mid \exists q \in F.\, (p, \epsilon, 1) \Rightarrow (q, w, 1) \,\}$$

We call $\mathcal{A}$ *ϵ-free* if δ does not contain tuples of the form (p, ϵ, n, q). Note that for any ϵ-free valence automaton there is an equivalent one that only contains single letters in its transitions, for every transition $(p, a_1 a_2 \cdots a_n, m, q)$ can be replaced by transitions

$$(p, a_1, 1, p_1), (p_1, a_2, 1, p_2), \ldots, (p_{n-1}, a_n, m, q).$$

An ϵ-free valence automaton $\mathcal{A}$ in which every transition contains only single letters can be regarded as a $\mathbf{T}$-automaton (★) for the nondeterministic monoid action theory over M. To see this let $B = \mathcal{P}_\omega(M \times 1) \cong \mathcal{P}_\omega(M)$ be the free $\mathbf{T}$-algebra on 1 and define the map $o^m : X \to \mathcal{P}_\omega(M)$ by $o^m(q) = \{1\}$ if $q \in F$ and $o^m(q) = \emptyset$ else, and the transitions in δ yield the map $t^m : X \to \mathcal{P}_\omega(M \times X)^A$. Using Lemma 5.9 it is now not difficult to prove that $\{w \in A^* \mid 1 \in [\![q_0]\!]_m\}$ is the language accepted by $\mathcal{A}$.

Example 7.3 (*Nondeterministic polycyclic theory*). A relevant special case of the previous example is when M is a *polycyclic monoid* [Lawson 1999]. This means that M is the monoid over a set of generators $\lightning, g_1, \ldots, g_k, \ldots, g_1^{-1}, \ldots, g_k^{-1}$ and satisfying the relations

$$\lightning\, g_i = g_i \lightning = \lightning, \qquad g_i g_i^{-1} = 1, \qquad g_i g_j^{-1} = \lightning \quad (i \neq j).$$

The number k is called the *rank* of M. We call the theory of $\mathcal{P}_\omega(M \times (-))$ the *nondeterministic polycyclic theory*.

The technical distinction between the nondeterministic stack theory and the nondeterministic polycyclic theory is minor. On the one hand, the nondeterministic stack theory uses the zero 0 of the semiring to model failure in computing the right inverse, while the nondeterministic polycyclic theory has its own zero $\lightning$, which coexists with 0. On the other hand, emptiness detection is explicitly available for stacks (using e) but not for polycyclic monoids.

It is well-known that valence automata over polycyclic monoids of rank at least 2 recognize context-free languages and so do nondeterministic stack $\mathbf{T}$-automata. We would like to give a uniform proof of this fact applying both to Example 7.1 and to Example 7.3.

First, observe that if the semiring R is idempotent then any R-semimodule (equivalently, $\mathbf{T}_R$-algebra) B can be partially ordered by putting $b \leq c$ iff $b + c = c$.

Definition 7.4. Given a $\mathbf{T}_R$-automaton $m : X \to B \times (T_R X)^A$, and an initial state $x_0 \in X$ we define the language *recognized by* $b \in B$ by

$$L_b(m) = \{w \in A^* \mid [\![x_0]\!]_m(w) \geq b\}.$$

Note that both semirings arising from Examples 7.1 and 7.3 are idempotent (since addition is given by union of sets). For nondeterministic stack $\mathbf{T}$-automata we typically choose as $b \in B \subseteq 2^{\Gamma^*}$

the predicate distinguishing the initial stack configuration, e.g. $b(w) = \top$ iff w is the initial stack symbol. For valence automata over M we take $B = \mathcal{P}_\omega(M)$ and $b = \{1\}$. Then the above definition of accepted languages instantiates as expected.

Recall that the language of balanced parentheses, or *Dyck language* is a language $\mathcal{D}_n \subseteq \mathcal{A}_n = \{(_1,)_1, \dots, (_n,)_n\}^*$ consisting of string of parentheses balanced in the standard sense. The following result is a reformulation of the classical Chomsky-Schützenberger theorem.

THEOREM 7.5. *Let α be a monoid morphism from $\mathcal{A}_2$ to the multiplicative structure of some idempotent semiring R such that*

(1) *for some $b_0, b_1 \in B$, $\alpha(w) \cdot b_0 \geq b_1$ iff w is balanced;*
(2) *for any c_1, c_2 if $c_1 + c_2 \geq b_1$ then either $c_1 \geq b_1$ or $c_2 \geq b_1$.*

Then for any context-free language there is a T_R-automaton recognizing it by b_1.

PROOF. Let us denote $\Omega_n = \{(_1,)_1, \dots, (_n,)_n\}$ so that $\mathcal{A}_n = \Omega_n^*$. First note that from $\alpha : \mathcal{A}_2 \to R$ that we postulated we can obtain a monoid morphism $\alpha' : \mathcal{A}_n \to R$ for every n with the same property (1). Indeed, following [Book 1975], we define a monoid morphism $\gamma : \mathcal{A}_n \to \mathcal{A}_2$ sending every $(_n$ to $(_1^n(_2$ and every $)_n$ to $)_2)_1^n$ and having the property that $\mathcal{D}_n = \gamma^{-1}(\mathcal{D}_2)$, which means that if $\gamma(w) \in \mathcal{A}_2$ is balanced then w is also balanced. Since the converse is obvious, the composition $\alpha' = \alpha \cdot \gamma : \mathcal{A}_n \to R$ inherits from α the property that $\alpha'(w) \cdot b_0 \geq b_1$ iff w is balanced.

Let $\mathcal{L}$ be any context-free language. By Theorem 4.13 and Proposition 5.5, it suffices to prove that there is an additive reactive expression e such that

$$\mathcal{L} = \{w \in A^* \mid [\![e]\!](w) \geq b_1\}.$$

By the Chomsky-Schützenberger theorem, we have $\mathcal{L} = \beta(\mathcal{R} \cap \mathcal{D}_n)$ for some regular language $\mathcal{R}$ over Ω_n and some monoid morphism $\beta : \mathcal{A}_n \to A^*$, and, according to the above argument, in what follows let us regard α as a morphism from $\mathcal{A}_n$ to R. We use the version of the Chomsky-Schützenberger theorem from [Okhotin 2012] where it is shown that if $\mathcal{L}$ does not contain one-letter words then β can be chosen *non-erasing*, i.e. $\epsilon \notin \beta(g)$ for all $g \in \Omega_n$. The assumption that $\mathcal{L}$ does not contain one-letter words does not restrict generality, for if we could show for $\mathcal{L}' = \mathcal{L} \setminus A$ and an expression e that $\mathcal{L}' = \{w \in A^* \mid [\![e]\!](w) \geq b_1\}$ then of course we would have

$$\mathcal{L} = \left\{w \in A^* \mid [\![e + \sum\nolimits_{a \in \mathcal{L} \cap A} a.b_1]\!](w) \geq b_1\right\}.$$

Henceforth we assume that $\mathcal{L} \cap A = \emptyset$ and β is non-erasing.

As we know from Propositions 4.7 and 5.5, $\mathcal{R}$ can be given by an additive reactive expression over the boolean semiring $R = B = \{0, 1\}$. We replace in this expression every occurrence of the form $g.-$ where $g \in \Omega_n$ by $a_1. \dots a_k. \alpha(g) \cdot (-)$ where $a_1 \cdots a_k = \beta(g)$ and every occurrence of $1 \in B_0 = \{1\}$ by b_0. The resulting expression e is a reactive additive expression for the semimodule monad T_R. Note that the assumption that β is nonerasing ensures that e remains guarded. It is then easy to check that

$$[\![e]\!](w) \geq r \cdot b_0 \quad \text{if} \quad \exists u \in \mathcal{A}_n.\, r = \alpha(u) \wedge w = \beta(u); \tag{7.1}$$

indeed, given $u = g_1 \dots g_k \in \mathcal{A}_n$ with $g_i \in \Omega_n$ such that $r = \alpha(u)$ and $w = \beta(u)$, by definition, $[\![e]\!](w) \geq r \cdot b_0$ iff $o(\partial_{\beta(g_1)\cdots\beta(g_k)}(e)) \geq \alpha(g_1) \cdots \alpha(g_k) \cdot b_0$, which follows from the definition of e, specifically, from the way $g.-$ is replaced. Suppose, $w \in \mathcal{L} = \beta(\mathcal{R} \cap \mathcal{D}_n)$. Then there is $u \in \mathcal{D}_n$ such that $w = \beta(u)$. By assumption (1), $\alpha(u) \cdot b_0 \geq b_1$ and by (7.1), $[\![e]\!](w) \geq \alpha(u) \cdot b_0$. Therefore, $[\![e]\!](w) \geq b_1$.

For the converse, suppose $[\![e]\!](w) \geq b_1$ and show that $w \in \mathcal{L}$. Note that $[\![e]\!](w)$ is representable as a finite sum $\alpha(u_1) \cdot b_0 + \cdots + \alpha(u_k) \cdot b_0$ in such a way that $w = \beta(u_i)$ and $u_i \in \mathcal{R}$ for all i. By assumption (2), $\alpha(u_j) \cdot b_0 \geq b_1$ for some j and therefore by assumption (1), u_j is balanced. Since $w = \beta(u_j)$, $u_j \in \mathcal{R}$ and $u_j \in \mathcal{D}_n$, we obtain $w \in \mathcal{L}$. □

Example 7.6. Let us check that conditions of Theorem 7.5 apply to Examples 7.1 with $|\Gamma| > 1$ and 7.3 with $k > 1$.

1. For the nondeterministic stack theory, let us consider $B \subseteq 2^{\Gamma^*} \cong \mathcal{P}(\Gamma^*)$ as in Definition 6.5 (for $m = 1$). It is not difficult to work out that the action of R on B satisfies for every given $f : \Gamma^* \to 2$ the following laws

$$e \cdot f(u) = \begin{cases} f(\epsilon) & \text{if } u = \epsilon \\ 0 & \text{else,} \end{cases} \qquad o_i \cdot f(u) = \begin{cases} f(v) & \text{if } u = \gamma_i v \\ 0 & \text{else,} \end{cases} \qquad u_i \cdot f(u) = f(\gamma_i u); \tag{7.2}$$

in fact, this holds because $e \cdot (-)$, $o_i \cdot (-)$ and $u_i \cdot (-)$ are the unary operations $empty^B$, pop_i^B and $push_i^B$, respectively, by using the definition of $empty$, pop_i and $push_i$ from $push$ and pop (see Example 7.1), and by using how $pop^B : B^{n+1} \to B$ and $push^B : B \to B$ act ensuing the definition of the algebra structure α^m on B (see Definitions 6.5, and 5.7).

We take $\alpha : \mathcal{A}_2 \to R$ sending $(_i$ to u_i, $)_i$ to o_i for $i = 1, 2$ and $b_0 = b_1 = \{\epsilon\}$. Condition (2) holds because $\{\epsilon\}$ is an atom of the Boolean algebra $\mathcal{P}(\Gamma^*)$ (noting that $+$ on B is union of languages over Γ). Condition (1) means that w is balanced iff $\alpha(w) \cdot \{\epsilon\} \supseteq \{\epsilon\}$. This is easy to verify: on the one hand, if w is balanced, then $\alpha(w)$ can be reduced to 1 by successively replacing every $\alpha((_i)_i) = u_i o_i$ by 1, and therefore for such w, $\alpha(w) \cdot \{\epsilon\} = \{\epsilon\}$; on the other hand, if w is not balanced, by replacing $\alpha((_i)_i) = u_i o_i$ with 1 we eventually obtain that $\alpha(w)$ either (i) contains a factor $u_i o_j$ with $i \neq j$, or (ii) contains a factor $o_i u_j$, or (iii) is a nonempty product of u_i's, or (iv) is a nonempty product of o_i's. In the cases (i)–(iii), we see that $\alpha(w) \cdot \{\epsilon\}$ is $\emptyset$ using the relations from Example 7.1 and the equations in (7.2). In the remaining case, $\alpha(w)$ is a (nonempty) product of the o_i, and hence $\alpha(w) \cdot \{\epsilon\} \supseteq \{\epsilon\}$ would imply a contradiction: $(e\alpha(w)) \cdot \{\epsilon\} = 0 \cdot \{\epsilon\} = \emptyset \supseteq e \cdot \{\epsilon\} = \{\epsilon\}$.

2. For the polycyclic theory we take $\alpha : \mathcal{A}_2 \to \mathcal{P}_\omega(M)$ sending $(_i$ to $\{g_i\}$ and $)_i$ to $\{g_i^{-1}\}$ for $i = 1, 2$ and $b_0 = b_1 = \{1\}$. The verification of conditions (1) and (2) here is analogous, in particular, for (1) one readily checks that $\alpha(w) = 1$ iff w is balanced.

Contrasting [Kambites 2009] we cannot replace the polycyclic monoid in Example 7.3 by a free group and conclude by Theorem 7.5 that automata with free group memory recognize context-free languages because the relevant construction would make an essential use of internal transitions, which we do not allow.

As we have seen by Examples 7.1 and 7.2, for any Σ-theory we can automatically generate its nondeterministic variant by tensoring with $\mathcal{E}_{\mathcal{P}_\omega}$ and this has a sensible interpretation in terms of $\mathbf{T}$-automata. One may wonder if it is possible to convert a given $\mathbf{T}$-automaton to a $\mathbf{T} \otimes \mathcal{P}_\omega$-automaton (which is necessarily a weighted $\mathbf{T} \otimes \mathcal{P}_\omega$-automaton, by Proposition 6.3) preserving the semantics. The answer turns out to be affirmative under a natural assumption on the $\mathbf{T}$-algebra component B.

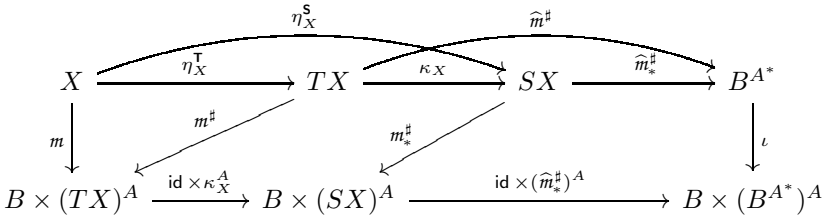
Let us first establish the following general result. It follows from [Bonsangue et al. 2015, Proposition 5.1]; we include a proof for the convenience of the reader.

LEMMA 7.7. *Let $\kappa : \mathbf{T} \to \mathbf{S}$ be a monad morphism, and let $m : X \to B \times TX^A$ and $m_* : X \to B \times SX^A$ be a $\mathbf{T}$- and an $\mathbf{S}$-automaton over X, respectively, such that*

$$o^m = o^{m_*}, \qquad \alpha^m = \alpha^{m_*} \cdot \kappa_B, \qquad \kappa_X \cdot t^m = t^{m_*}$$

(in particular this implies that B is simultaneously a $\mathbf{T}$- and an $\mathbf{S}$-algebra). Then the language semantics of m and m_* agree, i.e. $[\![x]\!]_m = [\![x]\!]_{m_*}$ for any $x \in X$.

PROOF. The proof amounts to showing commutativity of the following diagram:



The left-hand triangle commutes by the definition of $m^\sharp$, the right-hand part by the finality of B^{A^*} (recall from (4.1) that $\widehat{m}^\sharp_*$ denotes the unique coalgebra morphism) and the upper left-hand triangle since κ is a monad morphism. The upper right-hand triangle commutes by uniqueness of the final map from TX to B^{A^*} as soon as we establish commutativity of the middle parallelogram.

To see the latter we will use the freeness of the $\mathbf{T}$-algebra (TX, μ_X) (see Section 3) in the upper left-hand corner, i.e. we shall show that all morphisms in this part are $\mathbf{T}$-algebra morphisms and that this part commutes when precomposed with η^{T}_X. Indeed, the latter follows from the fact that the upper left-hand triangle commutes and since clearly

$$m_* = \big(X \xrightarrow{\ m\ } B \times (TX)^A \xrightarrow{\ \mathsf{id} \times (\kappa_X)^A\ } B \times (SX)^A\big).$$

Now to see that all morphisms in the middle part are $\mathbf{T}$-algebra morphism, recall first that the monad morphism $\kappa : \mathbf{T} \to \mathbf{S}$ induces a functor $\bar{\kappa}$ from the category of $\mathbf{S}$-algebras to the category of $\mathbf{T}$-algebras given on objects by $(Y, t) \mapsto (Y, t \cdot \kappa_Y)$ and being identity on morphism. Clearly, this functor maps (B, α^{m_*}) to (B, α^m). Now let β and β^*, denote the algebraic structures on $B \times (TX)^A$ and $B \times (SX)^A$, respectively, which are componentwise given by the structures of the free algebras (TX, μ_X) and by α^m and α^{m_*} on B, respectively. Now consider the four morphisms of the middle parallelogram of our diagram: (1) $\kappa_X : TX \to SX$ is easily seen to be a $\mathbf{T}$-algebra morphism from the free $\mathbf{T}$-algebra (TX, μ^{T}_X) to the $\mathbf{T}$-algebra $(SX, \mu^{\mathsf{S}}_X \cdot \kappa_{SX})$ (since κ is a monad morphism) and therefore (2) $\mathsf{id} \times (\kappa_X)^A$ is a $\mathbf{T}$-algebra morphism from $(B \times (TX)^A, \beta)$ to $(B \times (T_B X)^A, \beta^* \cdot \kappa_{B \times (SX)^A})$; (3) $m^\sharp$ is by definition a $\mathbf{T}$-algebra morphism and (4) $m^\sharp_*$ is an $\mathbf{S}$-algebra morphism and hence by applying the functor $\bar{\kappa}$ we see it is also a $\mathbf{T}$-algebra morphism as desired. □

We immediately obtain the following corollary.

COROLLARY 7.8. *Let B be a $\mathbf{T} \otimes \mathcal{P}_\omega$-algebra with structure $\alpha^{m_*} : (T \otimes \mathcal{P}_\omega)B \to B$. Then B is also a $\mathbf{T}$-algebra under $\alpha^{m_*} \cdot \kappa_B : TB \to B$ where $\kappa : \mathbf{T} \to \mathbf{T} \otimes \mathcal{P}_\omega$ is the left tensor injection. Let m be any $\mathbf{T}$-automaton (★) with $\alpha^m = \alpha^{m_*} \cdot \kappa_B$ and form the $\mathbf{T} \otimes \mathcal{P}_\omega$-automaton m_* with $o^{m_*} = o^m$, $t^{m_*} = \kappa_X \cdot t^m$ and the given α^{m_*}. Then the semantics of m and m_* agree; in symbols: $[\![x]\!]_m = [\![x]\!]_{m_*}$ for every state $x \in X$.*

Effectively, Corollary 7.8 states that for every $\mathbf{T}$ we can understand a $\mathbf{T}$-automaton as a special non-deterministic automaton, i.e. a $\mathbf{T} \otimes \mathcal{P}_\omega$-automaton, provided that its output $\mathbf{T}$-algebra B additionally carries the structure of a commutative idempotent monoid which commutes with the operations of $\mathbf{T}$ (in the sense of satisfying the tensor laws). In particular, this works well with submonads $\mathbf{T}$ of the state monad over a store S and output algebras B which are subalgebras of 2^S (see e.g. Example 5.7).

8. CPS-TRANSFORMS OF T-AUTOMATA AND R.E.-LANGUAGES

Theorem 6.7 suggests that the present language semantics is unlikely to produce languages beyond $\mathsf{NTIME}(n)$ under a computationally convincing choice of the components of a $\mathbf{T}$-automaton (★). The approach suggested by the classical formal language theory is to replace A with the set $A_\tau =$

$A \cup \{\tau\}$, where τ is a new *unobservable action*[2], but in lieu of the formal power series $B^{A^*_\tau}$ we use B^{A^*} as the semantic domain. This new *observational semantics* is supposed to be obtainable from the standard one by eliminating the unobservable actions.

We argue briefly, why the general assumptions about the structure of a $\mathbf{T}$-automaton are not sufficient to define the observational semantics. Consider the Moore automaton $m : X \to B \times X^{A_\tau}$ with $A = \{a\}$ and $B = \{b_0, b_1\}$. The underlying monad is the identity monad and α^m is the identity morphism. Let $X = \{x_0, x_1\}$, $o_m = \{\langle x_0, b_0\rangle, \langle x_1, b_1\rangle\}$, $t_m = \{\langle x_0, a, x_0\rangle, \langle x_0, \tau, x_1\rangle, \langle x_1, a, x_1\rangle, \langle x_1, \tau, x_1\rangle\}$. Removal of τ-transitions leads to a nondeterministic automaton having two a-transitions from x_0 to states marked with b_0 and with b_1 by o_m, which cannot be determinized unless we assume the structure of a commutative idempotent monoid (i.e. a $\mathcal{P}_\omega$-algebra structure) on B. A similar argument applied to looped internal transitions shows that B must support countable iterations of the monoid operation.

Using these assumptions on B, our idea is to use Lemma 7.7 to transform a given $\mathbf{T}$-automaton m to some $\mathbf{S}$-automaton m_* for which τ-transitions can be eliminated in a generic way. After eliminating the τ-transitions we then obtain an $\mathbf{S}$-automaton m_v, and finally we define the observational semantics of m as the standard language semantics of m_v. Note that Corollary 7.8 does not offer a sufficiently good candidate for m_*, because we would have to assume that B is a $\mathbf{T} \otimes \mathcal{P}_\omega$-algebra, which would rule out too many interesting instances, e.g. Examples 5.2 and 5.3. We therefore obtain m_* by a technique borrowed from higher-order programming language semantics and known as *continuation passing style (CPS)* transformation [Plotkin 1975].

Let $\alpha : TB \to B$ be a $\mathbf{T}$-algebra. We denote by $\mathbf{T}_B$ the continuation monad (see Example 3.10) with $T_B X = B^{B^X}$. We define $\kappa : \mathbf{T} \to \mathbf{T}_B$ by sending $p \in TX$ to $\kappa_X(p) = \lambda f.\, (\alpha \cdot Tf(p)) \in T_B X$, which yields a monad morphism; in fact, it is well known that for every monad $\mathbf{T}$ on a category with powers the above assignment of κ to α is part of a bijective correspondence between Eilenberg-Moore algebras on B and monad morphisms from $\mathbf{T}$ to $\mathbf{T}_B$ (see e.g. [Kock 1970, Theorem 3.2]).

Construction 8.1. Given a $\mathbf{T}$-automaton (★), let $\kappa : \mathbf{T} \to \mathbf{T}_B$ be the monad morphism given by $\kappa_X(p) = \lambda f.\, (\alpha^m \cdot Tf(p))$ and let

$$o^{m_*} = o^m, \qquad t^{m_*} = \kappa_X \cdot t^m, \qquad \alpha^{m_*} = \lambda t.\, t(\mathsf{id}),$$

which yields a $\mathbf{T}_B$-automaton[3] $m_* : X \to B \times (T_B X)^A$. It is easily seen that $\alpha^{m_*} : T_B B \to B$ is a $\mathbf{T}_B$-algebra and $\alpha^m = \alpha^{m_*} \cdot \kappa_B$. We call m_* the *CPS-transform* of (★).

The following is another a corollary of Lemma 7.7.

COROLLARY 8.2. *The language semantics of a $\mathbf{T}$-automaton and of its CPS-transform agree; more precisely, for every $\mathbf{T}$-automaton (★) and a state $x \in X$, $[\![x]\!]_m = [\![x]\!]_{m_*}$.*

Corollary 8.2 implies Proposition 4.7 announced previously in Section 4.

PROOF OF PROPOSITION 4.7. If B in (★) is finite then, by definition, $T_B X$ is also finite. Thus, the generalized powerset construction performed on the CPS-transform m_* yields a Moore automaton. Thus, for every $x \in X$, $[\![x]\!]_m = [\![x]\!]_{m^*}$ is a regular formal power series. □

We now proceed with the definition of the observational semantics. In order to do this we shall make use of algebras for the *countable multiset monad* $\mathbf{M}$. Its monad structure will not be needed; however, we recall its definition for the convenience of the reader.

Remark 8.3. For the countable multiset monad $\mathbf{M}$, MX consists of countable multisets on X, i.e.

$$MX = \{f : X \to \mathbb{N}_\infty \mid f \text{ has countable support}\},$$

[2] We prefer to use τ instead of the more standard ϵ to avoid confusion with the empty word.
[3] We abuse terminology here since $\mathbf{T}_B$ is not finitary (see Remark 4.4).

where $\mathbb{N}_\infty = \mathbb{N} + \{\infty\}$ with the operations of addition and multiplication extended to ∞ in the expected way. The unit of $\mathbf{M}$ is given by $\eta_X(x) = \delta_x : X \to \mathbb{N}_\infty$ with $\delta_x(x) = 1$ and $\delta_x(y) = 0$ otherwise. For any map $h : X \to MY$ the Kleisli lifting $h^\star : MX \to MY$ acts as follows:

$$h^\star(f)(y) = \sum\nolimits_{x \in X} f(x) \cdot h(x)(y).$$

An $\mathbf{M}$-algebra is, equivalently, a commutative monoid with infinite summation satisfying the expected laws. We call such a monoid *ω-additive*. For an ω-additive monoid we denote by $\emptyset$ the neutral element, by $a + b$ the binary sum and by $\sum_{i=1}^{\infty} a_i$ the countable sum.

Definition 8.4 (*ω-additive $\mathbf{T}$-automata*). A $\mathbf{T}$-automaton ($\bigstar$) is *ω-additive* if B (besides being $\mathbf{T}$-algebra) is an ω-additive monoid.

It is easy to see that the ω-additive monoid structure extends from B to $T_B X$ pointwise:

LEMMA 8.5. *If B is an ω-additive monoid and a $\mathbf{T}$-algebra then for every set X, $T_B X$ is an ω-additive monoid.*

PROOF. This follows from the fact that B carries an Eilenberg-Moore algebra structure for the countable multiset monad $\mathbf{M}$. Equivalently, we have a monad morphism $m : \mathbf{M} \to \mathbf{T}_B$ (see [Kock 1970, Theorem 3.2]). Thus, by forming $(T_B X, \mu_X^{\mathbf{T}_B} \cdot m_{T_B X})$ we obtain an Eilenberg-Moore algebra structure for $\mathbf{M}$ on $T_B X$, i.e., $T_B X$ is an ω-additive monoid. □

The ω-additive monoid structure on $T_B X$ allows us to define for any given $\mathbf{T}$-automaton over the alphabet A_τ a $\mathbf{T}_B$-automaton over A. To this end, we first form the CPS-transform of the given $\mathbf{T}$-automaton and then use infinite summation to get rid of unobservable actions τ:

Construction 8.6. Given a $\mathbf{T}$-automaton $\mathcal{m} : X \to B \times (TX)^{A_\tau}$, we construct $\mathcal{m}_v : X \to B \times (T_B X)^A$ with $\alpha^{\mathcal{m}_v} = \alpha^{\mathcal{m}_*} = \lambda t.\, t(\mathsf{id})$ and with $t^{\mathcal{m}_v}$, $o^{\mathcal{m}_v}$ defined as

$$t^{\mathcal{m}_v}(x_0, a) = \sum\nolimits_{i=1}^{\infty} \mathsf{do}\ x_1 \leftarrow t^{\mathcal{m}_*}(x_0, \tau); \ldots; x_{i-1} \leftarrow t^{\mathcal{m}_*}(x_{i-2}, \tau); t^{\mathcal{m}_*}(x_{i-1}, a),$$
$$o^{\mathcal{m}_v}(x_0) = o^{\mathcal{m}_*}(x_0) + \sum\nolimits_{i=1}^{\infty} \big(\mathsf{do}\ x_1 \leftarrow t^{\mathcal{m}_*}(x_0, \tau); \ldots; t^{\mathcal{m}_*}(x_{i-1}, \tau)\big)(o^{\mathcal{m}_*}).$$

(Note that $\mathsf{do}\ x_1 \leftarrow t^{\mathcal{m}_*}(x_0, \tau); \ldots; t^{\mathcal{m}_*}(x_{i-1}, \tau)$ is an element of $T_B X = B^{B^X}$, i.e. a function $B^X \to B$ which can be applied to $o^{\mathcal{m}_*} \in B^X$.)

Intuitively, for $t^{\mathcal{m}_v}(x_0, a)$ we accumulate the effects underlying the τ-transitions preceding the first a-transition; for $o^{\mathcal{m}_v}(x_0)$ we accumulate the effects along any sequence of τ-transition leading to an accepting state detected by $o^{\mathcal{m}_*}$.

We define the observational semantics for $\mathcal{m}$ to be the language semantics for $\mathcal{m}_v$.

Definition 8.7 (*Observational semantics*). Given a $\mathbf{T}$-automaton ($\bigstar$) over input alphabet A_τ, its *observational semantics* is defined as $[\![-]\!]^\tau_{\mathcal{m}} = [\![-]\!]_{\mathcal{m}_v}$.

In order to consider the observational semantics $[\![-]\!]^\tau_{\mathcal{m}}$ in concrete instances of $\mathbf{T}$-automata, we need a description similar to Lemma 5.9. Before we state and prove it we make some auxiliary observations.

Remark 8.8. Since $\kappa : \mathbf{T} \to \mathbf{T}_B$ is a monad morphism we have that for every $f : X \to TY$:

$$\kappa_Y\,(\mathsf{do}\ x \leftarrow p; f(x)) = \mathsf{do}\ x \leftarrow \kappa_X(p); \kappa_Y \cdot f(x). \tag{8.1}$$

Remark 8.9. We shall need two properties of the ω-additive monoid structure on $T_B X$.

1. Kleisli substitution distributes over sums:

$$\mathsf{do}\ y \leftarrow \sum\nolimits_{i=1}^{\infty} p_i; f(y) = \sum\nolimits_{i=1}^{\infty} \mathsf{do}\ y \leftarrow p_i; f(y). \tag{8.2}$$

Indeed, this equation expresses that the outside of the following diagram commutes for every $f : X \to T_B Y$ (here we abbreviate T_B as T, and recall that M denotes the countably supported multiset monad):

$$\begin{array}{ccccc} MTX & \xrightarrow{m_X} & TTX & \xrightarrow{\mu_X} & TX \\ {\scriptstyle Mf^\star}\downarrow & & {\scriptstyle Tf^\star}\downarrow & & \downarrow{\scriptstyle f^\star} \\ MTY & \xrightarrow[m_Y]{} & TTY & \xrightarrow[\mu_Y]{} & TY \end{array}$$

And this diagram clearly commutes by the naturality of the monad morphism $m : \mathbf{M} \to \mathbf{T}_B$, and since $f^\star$ is a $\mathbf{T}$-algebra morphism.

2. Similarly, sums commute with the $\mathbf{T}_B$-algebra structure α^{m_*}, i.e. the following equation holds for every countable family of elements $p_i \in T_B B$:

$$\alpha^{m_*}\Big(\sum\nolimits_{i=1}^{\infty} p_i\Big) = \sum\nolimits_{i=1}^{\infty} \alpha^{m_*}(p_i); \tag{8.3}$$

in other words, $\alpha^{m_*} : T_B B \to B$ is a morphism of ω-additive monoids. Indeed, this follows from the commutativity of the following diagram (again we abbreviate T_B by T):

$$\begin{array}{ccccc} MTB & \xrightarrow{m_{TB}} & TTB & \xrightarrow{\mu_B} & TB \\ {\scriptstyle M\alpha^{m_*}}\downarrow & & {\scriptstyle T\alpha^{m_*}}\downarrow & & \downarrow{\scriptstyle \alpha^{m_*}} \\ MB & \xrightarrow[m_B]{} & TB & \xrightarrow[\alpha^{m_*}]{} & B \end{array}$$

LEMMA 8.10. *Given a $\mathbf{T}$-automaton (★), $x_0 \in X$ and $u \in A^*$ then*

$$[\![x_0]\!]^\tau_m(\epsilon) = o^m(x_0) \; + \sum\nolimits_{i=1}^{\infty} \alpha^m\left(\mathsf{do}\ x_1 \leftarrow t^m(x_0,\tau); \ldots; x_i \leftarrow t^m(x_{i-1},\tau); \eta^{\mathbf{T}}_B \cdot o^m(x_i)\right)$$

$$[\![x_0]\!]^\tau_m(au) = \sum\nolimits_{i=1}^{\infty} \alpha^m\left(\mathsf{do}\ x_1 \leftarrow t^m(x_0,\tau); \ldots; x_i \leftarrow t^m(x_{i-1},a); \eta^{\mathbf{T}}_B \cdot [\![x_i]\!]^\tau_m(u)\right)$$

PROOF. We proceed by induction over the argument $w \in A^*$ of $[\![x_0]\!]^\tau_m$. For $w = \epsilon$:

$$\begin{aligned} [\![x_0]\!]^\tau_m(\epsilon) &= [\![x_0]\!]_{m_v}(\epsilon) && \text{// definition of } [\![-]\!]^\tau_m \\ &= o^{m_v}(x_0) && \text{// Lemma 5.9} \\ &= o^{m_*}(x_0) + \textstyle\sum_{i=1}^{\infty} \left(\mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0,\tau); \ldots; t^{m_*}(x_{i-1},\tau)\right)(o^{m_*}) \\ & && \text{// definition of } o^{m_v} \\ &= o^m(x_0) + \textstyle\sum_{i=1}^{\infty} \kappa_X\big(\mathsf{do}\ x_1 \leftarrow t^m(x_0,\tau); \ldots; t^m(x_{i-1},\tau)\big)(o^m) \\ & && \text{// repeated application of (8.1)} \\ & && \text{// with } o^{m_*} = o^m,\ t^{m_*} = \kappa_X \cdot t^m \\ &= o^m(x_0) + \textstyle\sum_{i=1}^{\infty} (\alpha^m \cdot T o^m)\big(\mathsf{do}\ x_1 \leftarrow t^m(x_0,\tau); \ldots; t^m(x_{i-1},\tau)\big) \\ & && \text{// definition of } \kappa_X \\ &= o^m(x_0) + \textstyle\sum_{i=1}^{\infty} \alpha^m\big(\mathsf{do}\ x_1 \leftarrow t^m(x_0,\tau); \ldots; x_i \leftarrow t^m(x_{i-1},\tau); \eta^{\mathbf{T}}_B \cdot o^m(x_i)\big) \\ & && \text{// property of do-notation} \end{aligned}$$

For the induction step we consider $w = au$ and compute:

$$\begin{aligned} [\![x_0]\!]^\tau_m(au) &= [\![x_0]\!]_{m_v}(au) && \text{// definition of } [\![-]\!]^\tau_m \\ &= \alpha^{m_v}\big(\mathsf{do}\ y \leftarrow t^{m_v}(x_0,a); \eta^{\mathbf{T}}_B \cdot [\![y]\!]_{m_v}(u)\big) \end{aligned}$$

$$
\begin{aligned}
&\qquad\qquad\qquad\qquad\qquad\qquad \text{// Lemma 5.9}\\
&= \alpha^{m_*}\big(\mathsf{do}\ x_i \leftarrow \big(\textstyle\sum_{i=1}^{\infty} \mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0,\tau);\ldots;t^{m_*}(x_{i-1},a)\big);\eta_B^{\mathbf{T}B}\cdot[\![x_i]\!]_{m_v}(u)\big)\\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{// definition of } t^{m_v}\text{, since } \alpha^{m_v}=\alpha^{m_*},\\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{// and renaming } y \text{ to } x_i\\
&= \alpha^{m_*}\big(\textstyle\sum_{i=1}^{\infty} \mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0,\tau);\ldots;x_i \leftarrow t^{m_*}(x_{i-1},a);\eta_B^{\mathbf{T}B}\cdot[\![x_i]\!]_{m_v}(u)\big)\\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{// (8.2)}\\
&= \alpha^{m_*}\big(\textstyle\sum_{i=1}^{\infty} \kappa_B\big(\mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0,\tau);\ldots;x_i \leftarrow t^{m_*}(x_{i-1},a);\eta_B^{\mathbf{T}}\cdot[\![x_i]\!]_{m_v}(u)\big)\big)\\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{// (8.1) and since } \kappa\cdot\eta^{\mathbf{T}}=\eta^{\mathbf{T}B}\\
&= \textstyle\sum_{i=1}^{\infty} \alpha^{m_*}\cdot\kappa_B\big(\mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0,\tau);\ldots;x_i \leftarrow t^{m_*}(x_{i-1},a);\eta_B^{\mathbf{T}}\cdot[\![x_i]\!]_{m_v}(u)\big)\\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{// (8.3)}\\
&= \textstyle\sum_{i=1}^{\infty} \alpha^{m}\big(\mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0,\tau);\ldots;x_i \leftarrow t^{m_*}(x_{i-1},a);\eta_B^{\mathbf{T}}\cdot[\![x_i]\!]_{m_v}(u)\big)\\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{// since } \alpha^{m_*}\cdot\kappa_B=\alpha^{m}\\
&= \textstyle\sum_{i=1}^{\infty} \alpha^{m}\big(\mathsf{do}\ x_1 \leftarrow t^{m}(x_0,\tau);\ldots;x_i \leftarrow t^{m}(x_{i-1},a);\eta_B^{\mathbf{T}}\cdot[\![x_i]\!]_{m}^{\tau}(u)\big)\\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{// definition of } [\![-]\!]_{m}^{\tau}. \qquad \square
\end{aligned}
$$

Example 8.11. We consider two concrete instances of our observational semantics.

1. Nondeterministic stack $\mathbf{T}$-automata m over A_τ, i.e. where $\mathbf{T}$ is the tensor product of the stack monad and $\mathcal{P}_\omega$, are in bijective correspondence with ordinary pushdown-automata (i.e. nondeterministic ones with ϵ-transitions). In fact, a similar construction to the one performed in the proof of Theorem 6.7 allows one to obtain for any given m, $x_0 \in X$ and $\gamma_0 \in \Gamma$ a push-down automaton that accepts the language

$$\{w \in A^* \mid [\![x_0]\!]_m^\tau(w)(\gamma_0) = \top\}.$$

Conversely, every pushdown automaton M yields a nondeterministic stack $\mathbf{T}$-automaton such that the above language is the language accepted by M. It follows that the class of these languages is precisely the class of context-free languages over A.

2. Coming back to Example 7.2 let us consider valence automata again, but now *with* ϵ-transitions. Given any valence automaton $\mathcal{A} = (X, M, A, \delta, q_0, F)$ we can again assume w.l.o.g. that its transitions are labelled with a single letter or ϵ. Then we can regard $\mathcal{A}$ as a $\mathbf{T}$-automaton m over A_τ for the nondeterministic monoid action theory over M. Using Lemma 8.10 it is not difficult to prove that $\{w \in A^* \mid 1 \in [\![q_0]\!]_m^\tau\}$ is the language accepted by $\mathcal{A}$.

We now proceed to define a class of $\mathbf{T}$-automata that correspond to classical Turing machines in the sense that the observational semantics yields precisely all recursively enumerable languages.

Definition 8.12 (*Tape* $\mathbf{T}$*-automaton*). A *tape* $\mathbf{T}$*-automaton* is a $\mathbf{T}$-automaton (★) for which

— $\mathbf{T}$ is the tape monad over Γ (see Definition 3.17);
— B is the set of predicates over $\mathbb{Z}\times\Gamma^{\mathbb{Z}}$ consisting of all those $p \in 2^{\mathbb{Z}\times\Gamma^{\mathbb{Z}}}$ for each of which there is a k such that $p(i,\sigma) = p(i,\sigma')$ and $p(i,\sigma_{+j}) = p(i+j,\sigma)$ whenever $\sigma \equiv \sigma' \pmod{[i-k, i+k]}$;
— $\alpha^m : TB \to B$ is given by evaluation; it restricts the morphism $T(2^S) = (2^S \times S)^S \xrightarrow{\mathsf{ev}^S} 2^S$, where $S = \mathbb{Z}\times\Gamma^{\mathbb{Z}}$.

The argument showing that B is indeed a $\mathbf{T}$-algebra is completely analogous to the one for stack $\mathbf{T}$-automata (Definition 5.7).

Tape $\mathbf{T}$-automata over A_τ are essentially deterministic 2-tape Turing machines with input alphabet A, where the first tape is a special read-only and right-only tape holding the input word at

the beginning of a computation. Thus, we obtain that tape automata represent all the recursively enumerable languages.

THEOREM 8.13. *For every tape* $\mathbf{T}$*-automaton* $\mathfrak{m}$ *over* A_τ, Γ *with* $|\Gamma| \geq 2$ *containing a special blank symbol* $\boxtimes$*, and every state* $x \in X$ *the following language is recursively enumerable:*

$$\{w \in A^* \mid [\![x]\!]^\tau_{\mathfrak{m}}(w)(0, \sigma_\boxtimes) = \top\},$$

where $\sigma_\boxtimes$ *is the constant function returning* $\boxtimes$*. Conversely, every recursively enumerable language can be represented in this way.*

In order to prove this theorem, we will relate tape automata and a special form of Turing machines called *online Turing machines*. The idea of an online Turing machine is a rather old one [Hennie 1966] and essentially amounts to equipping a standard (offline) Turing machine with an additional input tape which can only be read in one direction and not modified. From the coalgebraic point of view online Turing machines naturally extend finite state machines and push-down automata.

Definition 8.14 (*Online Deterministic Turing Machine (ODTM)*). An *online deterministic Turing machine* is a six-tuple $M = (Q, A, \Gamma, \delta, q_0, F)$, where Q is a finite set of states, A is the action (or input) alphabet, Γ is the tape alphabet (assumed to contain the special *blank* symbol $\boxtimes$), q_0 is the initial state, $F \subseteq Q$ is a set of final (or accepting) states and

$$\delta : Q \times (A \cup \{\tau\}) \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{N}, \mathsf{R}\}$$

is the transition function.

The difference to an ordinary TM is that transitions do not only depend on the tape contents but also on an input in the form of an action $a \in A$ given by the user from the outside during runtime of the machine, and there are also *internal* transitions, i.e. where a silent action τ triggers the transition. Hence, a *configuration* of an ODTM M is an element of

$$Q \times A^* \times (\mathbb{Z} \times \Gamma^{\mathbb{Z}})$$

consisting of the current state $q \in Q$ the remaining input actions $w \in A^*$ and a pair (i, σ) consisting of the current position i of the read/write head and tape content $\sigma : \mathbb{Z} \to \Gamma$. *Computations* (or *runs*) are then defined in the usual way as sequences of configurations starting from the initial configuration $(q_0, w, (0, \sigma_\boxtimes))$ where $w \in A^*$ is the input word and $\sigma_\boxtimes$ denotes the constant function on $\boxtimes$. Note that internal transitions leave the remaining input actions untouched while otherwise the head symbol is removed from $w \in A^*$ in a configuration.

Remark 8.15. The above definition is essentially the one from [Aanderaa 1974]. A nondeterministic variant of this definition has been recently employed by Baeten et al. [2011] under the name *reactive Turing machine* with the aim to equip TM's with a notion of interaction and so bridge the gap between classical computation and concurrency theory. In particular, the standard equivalence relation for reactive Turing machines is *bisimilarity* rather than *language equivalence* we study here.

Definition 8.16 (*Language of a ODTM*). Let M be an ODTM. The formal language accepted by M is the set of words $w \in A^*$ such that there exists a computation from the initial configuration to a configuration $(q, \epsilon, (n, \sigma))$ with $q \in F$.

More informally, a word is accepted by M if there is a computation that consumes all the letters in the input work w and leads to an accepting state. Note that due to the internal actions there may be several accepting computations of a word. So an ODTM is only deterministic in the sense that in every configuration there can be no two different moves consuming the same input letter. But an internal transition can happen nondeterministically in any configuration.

That ODTM's are an appropriate model of computations is stated by the following lemma.

LEMMA 8.17. *The class of languages accepted by ODTM's is the class of semi-decidable languages.*

PROOF. We show that an ordinary TM can be simulated by an ODTM and vice versa.

(a) Given an ODTM M it can be simulated by a nondeterministic TM $\bar{M}$ with two tapes as follows: the first (input) tape of $\bar{M}$ stores the input word $w \in A^*$ which is processed read-only from left to right, and the second tape of $\bar{M}$ corresponds to the tape of M. The NTM $\bar{M}$ simulates M as follows: in each step $\bar{M}$ nondeterministically either performs an internal action of M or reads one symbol from the first tape (then moving the head to the right by one position on this tape). In addition, $\bar{M}$ has a special accepting halting state q_f, and it can nondeterministically decide to move to that state from every accepting state of M whenever a blank symbol is read on the first tape; this allows $\bar{M}$ to halt and accept if M is in any accepting configuration after consuming its input. It is then clear that $\bar{M}$ and M accept the same language. We conclude that the language accepted by any ODTM is semi-decidable.

(b) Conversely, suppose we have a deterministic TM with input alphabet A. Then M can be simulated by an ODTM $\bar{M}$. The computation of $\bar{M}$ has two phases: in the first phase $\bar{M}$ consumes its entire input and writes it on its tape. During this phase no internal transitions happen. The first phase ends as soon as $\bar{M}$ performs its first internal action, which starts the second phase. In this phase $\bar{M}$ only performs internal actions in the sense that all transitions consuming an input symbol $a \in A$ lead to a non-accepting state that is never left again. At the beginning of the second phase $\bar{M}$ then moves the head to the first input symbol (if any) on its tape. It then starts a simulation of the DTM M using internal transitions only. Whenever M halts in a (non-)accepting state, then $\bar{M}$ moves to a (non-)accepting state that it never leaves again. Again, $\bar{M}$ clearly accepts the same language as M. Thus, it follows that every semi-decidable language is accepted by an ODTM. □

PROOF OF THEOREM 8.13. We give for a tape automaton m as in the statement of the theorem an equivalent ODTM and vice versa.

(a) Given m, we define an ODTM M. For every $x \in X$ and $a \in A$ let $k_{x,a}$ be the minimal natural number as in Definition 3.17 for

$$t^m(x,a) = \langle r, z, t\rangle \in TX.$$

Analogously, let l_x be the minimal natural number according to the second clause of Definition 8.12 for

$$o^m(x) : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to 2.$$

The state set of M consists of the states X of m times a finite memory that can store a finite portion of M's tape and is of the form

$$\{-n, \dots, 0, \dots, n\} \times \Gamma^{2n+1}, \qquad \text{where } n = \max\{l_x, \max\{k_{x,a} \mid a \in A\}\}.$$

We say that a memory content $(0, \bar{\sigma})$ *restricts* $(i, \sigma) \in \mathbb{Z} \times \Gamma^{\mathbb{Z}}$ if $\bar{\sigma}(j) = \sigma(i + j)$ for all $j = -n, \dots, 0, \dots, n$. The final states of M are those states $x \in X$ together with memory contents $(0, \bar{\sigma})$ that restrict (i, σ) with $o^m(x)(i, \sigma) = \top$; that this is well-defined follows from Definition 3.17. We now describe informally how M simulates m. Since m can access several symbols from the tape at once we need to simulate transitions of m by several steps of M. These steps will make sure that the contents of M's finite memory always restricts its tape contents. Hence, a transition of m given by $t^m(x,a)(i,\sigma) = (x', i', \sigma')$ is simulated by the following steps of M (where M starts in state x with the memory contents $(0, \bar{\sigma})$ restricting M's tape content (i, σ)):

(1) M performs a transition that consumes the input letter a and changes the state to x' and the memory content to the appropriate value $(j, \bar{\sigma}')$ that reflects the values of i' and σ', i.e. $j = i' - i$ and $\bar{\sigma}'(\ell) = \sigma'(i + \ell)$ for every $\ell = -n, \dots, 0, \dots, n$ (this is possible by Definition 3.17);
(2) now M replaces the $2n+1$ tape cells around the current position of the read/write head according to $\bar{\sigma}'$ from the memory content and then the read/write head's position is changed according to j (this uses a finite number of additional auxiliary states);

(3) finally, the memory is overwritten with the $2n+1$ tape symbols around the new position of the read/write head so that the computation of the m-transition ends in state x' with a memory content $(0, \bar{\sigma})$ restricting the new tape content (i', σ').

Note that all the above points except (1) are realized by internal transitions of M.

Now we need to prove that M accepts a word $w \in A$ from the initial state x_0 (with memory content $(0, \sigma_\boxtimes)$ iff $[\![x_0]\!]^\tau_m(w)(0, \sigma_\boxtimes) = \top$. We will prove more generally that for every state x_0, we have $[\![x_0]\!]^\tau_m(w)(z_0, \sigma_0) = \top$ iff there exists an accepting M-computation from state x_0 starting with tape content (z_0, σ_0).

Before we proceed with the proof recall that the $\mathbf{T}$-algebra structure $\alpha^m : TB \to B$ is given by evaluation. It follows that for every map $f : X \to TB$ the uncurrying of $\alpha^m \cdot f : X \to B \subseteq 2^S$ is

$$X \times S \xrightarrow{f'} B \times S \subseteq 2^S \times S \xrightarrow{\mathsf{ev}} 2,$$

where $S = \mathbb{Z} \times \Gamma^{\mathbb{Z}}$ and ev is the evaluation map.

Now we prove the desired statement by induction on w. For the base case observe that, by Lemma 8.10, $[\![x_0]\!]^\tau_m(\epsilon)(z_0, \sigma_0) = \top$ iff $o^m(x_0)(i_0, \sigma_0) = \top$ or there exists an $i \geq 1$ such that

$$\alpha^m\big(\mathsf{do}\ x_1 \leftarrow t^m(x_0, \tau); \ldots; x_i \leftarrow t^m(x_{i-1}, \tau); \eta^{\mathbf{T}}_B \cdot o^m(x_i)\big)(z_0, \sigma_0) = \top.$$

In the first case x_0 is a final state of M and so the empty (0-step) computation of M is an accepting M-computation of ϵ. In the second case, let i be such that the above equation holds. Equivalently, the following morphism

$$X \times S \longrightarrow \cdots \longrightarrow X \times S \xrightarrow{o^m \times S} B \times S \subseteq 2^S \times S \xrightarrow{\mathsf{ev}} 2,$$

where the unlabelled arrows form the i-fold composition of the uncurrying of $t^m(-, \tau) : X \to (X \times S)^S$, maps (z_0, σ_0) to 1. So equivalently, we have $x_1, \ldots, x_i$ and tape configurations (z_k, σ_k), $1 \leq k \leq i$, such that $t^m(x_k, \tau)(z_k, \sigma_k) = (x_{k+1}, z_{k+1}, \sigma_{k+1})$ for all $0 \leq k < i$, and $o^m(x_i)(z_i, \sigma_i) = \top$. Equivalently, we have an M-computation that performs steps (1)–(3) above i times (simulating τ-steps of m) and ends in the accepting state x_i with tape content (z_i, σ_i).

In the induction step of our proof let $w = au$. By Lemma 8.10, we have $[\![x_0]\!]^\tau_m(au)(z_0, \sigma_0) = \top$ iff there exists an $i \geq 1$ such that

$$\alpha^m\big(\mathsf{do}\ x_1 \leftarrow t^m(x_0, \tau); \ldots; x_i \leftarrow t^m(x_{i-1}, a); \eta^{\mathbf{T}}_B \cdot [\![x_i]\!]^\tau_m(u)\big)(z_0, \sigma_0) = \top.$$

By a similar argument as in the base case, this is equivalent to the existence of states $x_1, \ldots, x_i$ and tape content (z_k, σ_k), $1 \leq k \leq i$, such that $t^m(x_k, \tau)(z_k, \sigma_k) = (x_{k+1}, z_{k+1}, \sigma_{k+1})$ for all $0 \leq k < i-1$, $t^m(x_{i-1}, a)(z_{i-1}, \sigma_{i-1}) = (x_i, z_i, \sigma_i)$ and $[\![x_i]\!]^\tau_m(u)(z_i, \sigma_i) = \top$. The last conditions corresponds, by induction hypothesis, bijectively to an accepting M-computation from state x_i with initial tape content (z_i, σ_i). And the rest corresponds bijectively to an M-computation that consists of i-iterations of steps (1)–(3) simulating $i-1$ many τ-steps and one a-step of the given tape automaton m starting in state x_0 with tape content (z_0, σ_0) and ending in state x_i with tape content (z_i, σ_i). Putting these two parts together, we obtain the desired bijective correspondence to an accepting M-computation from state x_0 with initial tape content (z_0, σ_0).

(b) Conversely, given an ODTM $M = (Q, A, \Gamma, \delta, q_0, F)$ we construct an equivalent tape automaton m. We take Q as the set of states and we let

$$o^m(q)(z, \sigma) = \top \iff q \in F \quad \text{and} \quad t^m(q, a)(z, \sigma) = (q', z', \sigma'),$$

where q' and (z', σ') are the state and the tape content, respectively, of M after performing an a-transition in state q with tape content (z, σ). For internal transitions, $t^m(q, \tau)$ is defined analogously.

We need to prove that M accepts a word $w \in A$ iff $[\![q_0]\!]^\tau_m(w)(0, \sigma_\boxtimes) = \top$. More generally, one proves that for every state q_0 and tape content (z_0, σ_0) of M one has $[\![q_0]\!]^\tau_m(w)(z_0, \sigma_0) = 1$ iff there exists an accepting M-computation from state q_0 with initial tape content (z_0, σ_0). This is proved by induction on w once again. The details are similar (but slightly easier) than in part (a) of our proof, and so we leave them as an easy exercise for the reader. □

9. CONCLUSIONS AND FUTURE WORK

In the present paper, we have presented the first steps towards a uniform theory of effectful state machines combining Moore automata with computational monads. We have given a coalgebraic account of several types of state machines with effects (such as manipulation of a store, their accepted languages and syntactic expressions to specify them). We have presented several results of our theory including a generic Kleene-style theorem (Theorem 4.13) and one-direction of a Chomsky-Schützenberger-style theorem (Theorem 7.5). We have also given the first treatment of Turing machines in a coalgebraic setting: the observational language semantics of tape automata yields precisely the recursively enumerable languages.

There are several possible directions for future work. A converse to Theorem 7.5 is of interest. In addition, we plan to derive a sound calculus of reactive expressions extending [Bonsangue et al. 2013] and explore the boundaries for completeness. Such a calculus will depend on the monad $\mathbf{T}$ and its algebra B; in fact, while currently we only need the signature Σ and the algebra B for our results, the axioms of the theory presenting $\mathbf{T}$ will become laws of the calculus. Note that completeness is only possible for specific choices of $\mathbf{T}$ and B, for it follows from Corollary 6.8 that for the nondeterministic stack theory and B from Definition 6.5 a finite complete axiomatization is not possible.

Another interesting point is to capture further language and complexity classes, such as the context-sensitive languages using $\mathbf{T}$-automata. Capturing various classes of machines under the umbrella of coalgebra will result in standard tools such as bisimulation proof methods becoming available for those classes of machines and their language semantics. Hence, further investigations into such proof principles are of interest.

Acknowledgements. We thank the anonymous reviewers for their very careful reading of our manuscript and for their suggestions to improve the presentation.

REFERENCES

Stål O. Aanderaa. 1974. On k-tape versus $(k-1)$-tape real time computation. *Complexity of Computation* 7 (1974), 75–96.

Jiří Adámek, Horst Herrlich, and George Strecker. 1990. *Abstract and concrete categories*. John Wiley & Sons Inc., New York. xiv+482 pages.

Jiří Adámek, Stefan Milius, and Jiří Velebil. 2006. Iterative Algebras at Work. *Math. Structures Comput. Sci.* 16, 6 (2006), 1085–1131.

Jos Baeten, Bas Luttik, and Paul Tilburg. 2011. Reactive Turing Machines. In *FCT'11*, Olaf Owe, Martin Steffen, and JanArne Telle (Eds.). LNCS, Vol. 6914. Springer-Verlag, 348–359.

Falk Bartels. 2004. *On generalized coinduction and probabilistic specification formats*. Ph.D. Dissertation. Vrije Universiteit Amsterdam.

Marcello M. Bonsangue, Helle Hvid Hansen, Alexander Kurz, and Jurriaan Rot. 2015. Presenting Distributive Laws. *Log. Methods Comput. Sci.* 11, 3:2 (2015), 23 pp.

Marcello M. Bonsangue, Stefan Milius, and Alexandra Silva. 2013. Sound and Complete Axiomatizations of Coalgebraic Language Equivalence. *ACM Trans. Comput. Log.* 14, 1, Article 7 (2013), 52 pages.

Marcello M. Bonsangue, Jan J. M. M. Rutten, and Joost Winter. 2012. Defining Context-Free Power Series Coalgebraically. In *CALCO 2012*. 20–39.

Ronavld V. Book. 1975. *On the Chomsky-Schützenberger Theorem*. Technical Report 33. Dept. of Computer Science, Yale University.

Ronald V. Book and Sheila A. Greibach. 1970. Quasi-Realtime Languages. *Math. Systems Theory* 4, 2 (1970), 97–111.

Janusz A. Brzozowski. 1964. Derivatives of Regular Expressions. *J. ACM* 11, 4 (1964), 481–494.

Dion Coumans and Bart Jacobs. 2013. Scalars, monads, and categories. In *Quantum physics and linguistics. A compositional, diagrammatic discourse.*, Chris Heunen; Mehrnoosh Sadrzadeh and Edward Grefenstette (Eds.). Oxford University Press, 184–216.

Bruno Courcelle. 1983. Fundamental properties of infinite trees. *Theor. Comput. Sci.* 25, 2 (1983), 95 – 169.

Fredrik Dahlqvist and Renato Neves. 2017. Program semantics as Kleisli representations. (2017). preprint; available at https://fredrikdahlqvist.files.wordpress.com/2015/08/dahlqvist_neves1.pdf.

M. Droste, W. Kuich, and H. Vogler (Eds.). 2009. *Handbook of weighted automata*. Springer.

Manfred Droste, Werner Kuich, and Heiko Vogler (Eds.). 2009. *Handbook of Weighted Automata*. Springer.

Samuel Eilenberg. 1974. *Automata, Languages, and Machines*. Pure and Applied Mathematics, Vol. A. Academic Press.

Marcelo P. Fiore, Eugenio Moggi, and Davide Sangiorgi. 2002. A fully abstract model for the π-calculus. *Inf. Comput.* 179, 1 (2002), 76–117.

Peter Freyd. 1966. Algebra valued functors in general and tensor products in particular. *Colloq. Math.* 14 (1966), 89–106.

Seymour Ginsburg and Michael A. Harrison. 1968. One-way Nondeterministic Real-time List-storage Languages. *J. ACM* 15, 3 (1968), 428–446.

Joseph A. Goguen, James W. Thatcher, Eric G. Wagner, and Jesse B. Wright. 1977. Initial Algebra Semantics and Continuous Algebras. *J. ACM* 24, 1 (1977), 68–95.

Sergey Goncharov. 2013. Trace Semantics via Generic Observations. In *CALCO 2013 (LNCS)*, Reiko Heckel and Stefan Milius (Eds.), Vol. 8089. 158–174.

Sergey Goncharov, Stefan Milius, and Alexandra Silva. 2014. Towards a Coalgebraic Chomsky Hierarchy. In *TCS'14*, Vol. 8705. Springer, 265–280.

Michael A. Harrison and Ivan M. Havel. 1972. On a Family of Deterministic Grammars. In *In Proc. ICALP 1972*. 413–441.

Juris Hartmanis. 1967. Context-free languages and Turing machine computations. In *Proc. Sympos. Appl. Math. 19*. 42–51.

Frederick C. Hennie. 1966. On-Line Turing Machine Computations. *IEEE Trans. on Electronic Computers* EC-15, 1 (1966), 35–44.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2001. *Introduction to Automata Theory, Languages, and Computation* (2nd ed.). Addison-Wesley.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Martin Hyland, Paul Blain Levy, Gordon D. Plotkin, and John Power. 2007. Combining algebraic effects with continuations. *Theor. Comput. Sci.* 375, 1-3 (2007), 20–40.

Bart Jacobs. 2006. A Bialgebraic Review of Deterministic Automata, Regular Expressions and Languages. In *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday (LNCS)*, K. Futatsugi, J.-P. Jouannaud, and J. Meseguer (Eds.), Vol. 4060. 375–404.

Bart Jacobs, Alexandra Silva, and Ana Sokolova. 2012. Trace Semantics via Determinization. In *CMCS'12*. LNCS, Vol. 7399. Springer, 109–129.

Mark Kambites. 2009. Formal Languages and Groups as Memory. *Communications in Algebra* 37, 1 (2009), 193–208.

Bartek Klin. 2011. Bialgebras for structural operational semantics: An introduction. *Theor. Comput. Sci.* 412, 38 (2011), 5043–5069.

Anders Kock. 1970. On Double Dualization Monads. *Math. Scand.* 27 (1970), 151–165.

Anders Kock. 1972. Strong Functors and Monoidal Monads. *Arch. der Mathematik* 23, 1 (1972), 113–120.

Mark V. Lawson. 1999. *Inverse Semigroups: The Theory of Partial Symmetries*. World Scientific Publishing Company.

William Lawvere. 1963. Functorial Semantics of Algebraic Theories. *Proc. Natl. Acad. Sci. USA* 50, 5 (1963), 869–872.

Ming Li. 1985. Simulating two pushdown stores by one tape in $O(n^{1.5}\sqrt{\log n})$ time. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*. 56–64.

Saunders MacLane. 1998. *Categories for the working mathematician* (2nd ed.). Springer.

Stefan Milius. 2010. A Sound and Complete Calculus for finite Stream Circuits. In *Proc. LICS 2010*. IEEE Computer Society, 449–458.

Stefan Milius, Dirk Pattinson, and Thorsten Wißmann. 2016. A New Foundation for Finitary Corecursion: The Locally Finite Fixpoint and its Properties. In *Proc. FoSSaCS 2016 (LNCS)*, Bart Jacobs and Christof Löding (Eds.), Vol. 9634. Springer, 107–125.

Eugenio Moggi. 1991. Notions of Computation and Monads. *Inf. Comput.* 93 (1991), 55–92.

Robert Myers. 2013. *Rational Coalgebraic Machines in Varieties: Languages, Completeness and Automatic Proofs*. Ph.D. Dissertation. Imperial College London.

Alexander Okhotin. 2012. Non-erasing Variants of the Chomsky–Schützenberger Theorem. In *Developments in Language Theory*, Hsu-Chun Yen and OscarH. Ibarra (Eds.). LNCS, Vol. 7410. Springer, 121–129.

Dirk Pattinson and Lutz Schröder. 2016. Program equivalence is coinductive. In *Proc. LICS 2016*. IEEE Computer Society.

Simon Peyton Jones (Ed.). 2003. *The Haskell 98 Language and Libraries: The Revised Report*. Vol. 13. 0–255 pages.

Gordon Plotkin and John Power. 2002. Notions of Computation Determine Monads. In *FoSSaCS'02 (LNCS)*, Vol. 2303. Springer, 342–356.

Gordon Plotkin and John Power. 2003. Algebraic Operations and Generic Effects. *Appl. Cat. Struct.* 11 (2003), 69–94.

Gordon D. Plotkin. 1975. Call-by-name, call-by-value and the λ-calculus. *Theor. Comput. Sci.* 1 (1975), 125–159.

John Power and Olha Shkaravska. 2004. From Comodels to Coalgebras: State and Arrays. In *CMCS'04 (ENTCS)*, Vol. 106. 297–314.

Michael O. Rabin. 1963. Probabilistic Automata. *Information and Control* 6, 3 (1963), 230–245.

M. O. Rabin and D. Scott. 1959. Finite Automata and Their Decision Problems. *IBM J. Res. Dev.* 3, 2 (April 1959), 114–125.

Elaine Render and Mark Kambites. 2009. Rational subsets of polycyclic monoids and valence automata. *Information and Computation* 207, 11 (2009), 1329 – 1339.

Grzegorz Rozenberg and Arto Salomaa (Eds.). 1997. *Handbook of formal languages, vol. 1: Word, Language, Grammar*. Springer-Verlag New York, Inc.

Jan J. M. M. Rutten. 2000. Universal Coalgebra: A Theory of Systems. *Theor. Comput. Sci.* 249 (2000), 3–80.

Jan J. M. M. Rutten. 2003. Behavioural Differential Equations: A Coinductive Calculus of Streams, Automata, and Power Series. *Theor. Comput. Sci.* 308, 1-3 (2003), 1–53.

Jacques Sakarovitch. 2009. *Elements of Automata Theory*. Cambridge University Press.

Roberto Segala. 1995. *Modelling and Verification of Randomized Distributed Real-Time Systems*. Ph.D. Dissertation. Massachusetts Institute of Technology.

Roberto Segala and Nancy A. Lynch. 1995. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing* 2, 2 (1995), 250–273.

Alexandra Silva. 2010. *Kleene coalgebra*. Ph.D. Dissertation. Radboud Univ. Nijmegen.

Alexandra Silva, Filippo Bonchi, Marcello Bonsangue, and Jan Rutten. 2013. Generalizing determinization from automata to coalgebras. *Log. Methods Comput. Sci.* 9, 1:9 (2013), 27 pp.

Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. 2011. Quantitative Kleene Coalgebras. *Inform. and Comput.* 209, 5 (2011), 822–849.

Alexandra Silva, Marcello M. Bonsangue, and Jan J. M. M. Rutten. 2010. Non-deterministic Kleene coalgebras. *Log. Methods Comput. Sci.* 6, 3:23 (2010), 39 pp.

Don Syme, Adam Granicz, and Antonio Cisternino. 2007. *Expert F#*. Apress.

Terese. 2003. *Term Rewriting Systems*. Cambridge Tracts in Theoretical Computer Science, Vol. 55. Cambridge University Press.

Daniele Turi and Gordon D. Plotkin. 1997. Towards a mathematical operational semantics. In *Proc. LICS 1997*. 280–291.

Daniele Varacca and Glynn Winskel. 2006. Distributing probability over non-determinism. *Math. Struct. Comput. Sci.* 16 (2006), 87–113.

Joost Winter. 2014. *Coalgebraic Characterizations of Automata-Theoretic Classes*. Ph.D. Dissertation. Radboud University Nijmegen.

Joost Winter, Marcello M. Bonsangue, and Jan J. M. M. Rutten. 2013. Coalgebraic Characterizations of Context-Free Languages. *Log. Methods Comput. Sci.* 9, 3:14 (2013), 39 pp.

Georg Zetzsche. 2016. *Monoids as Storage Mechanisms*. PhD thesis.