Towards a Coalgebraic Chomsky Hierarchy

Sergey Goncharov[1], Stefan Milius[1], and Alexandra Silva[2]

[1] Lehrstuhl für Theoretische Informatik, Friedrich-Alexander-Universität Erlangen-Nürnberg
[2] Radboud University Nijmegen and Centrum Wiskunde & Informatica, Amsterdam

Abstract. The Chomsky hierarchy plays a prominent role in the foundations of theoretical computer science relating classes of formal languages of primary importance. In this paper we use recent developments on coalgebraic and monad-based semantics to obtain a generic notion of a $\mathbb{T}$-*automaton*, where $\mathbb{T}$ is a monad, which allows the uniform study of various notions of machines (e.g. finite state machines, multi-stack machines, Turing machines, valence automata, weighted automata). We use the *generalized powerset construction* to define a generic (trace) semantics for $\mathbb{T}$-automata, and we show by numerous examples that it correctly instantiates for some known classes of machines/languages captured by the Chomsky hierarchy. Moreover, our approach provides new generic techniques for studying expressivity power of various machine-based models.

1 Introduction

In recent decades much interest has been drawn to studying generic abstraction devices not only formally generalizing various computation models and tools, but also identifying core principles and reasoning patterns behind them. An example of this kind is given by the notion of *computational monad* [26], which made an impact both on the theory of programming (as an organization tool for denotational semantics [10, 28]) and on the practice (e.g. being implemented as a programming language feature of Haskell [1] and F# [40]). Another example is given by the theory of *coalgebras* [34], which provides a uniform framework for concurrency theory and observational semantics of systems.

In this paper, we use previous work on monads and coalgebras to give a combined (bialgebraic) perspective of the classical automata theory as well as of some less standard models such as weighted automata and valence automata. This does not only provided a unifying framework to study various computational models but also yields new techniques to prove expressivity bounds.

We base our framework on the notion of a $\mathbb{T}$-*automaton*, i.e. a coalgebra of the form

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

where T is the functor part of a monad $\mathbb{T}$, which we understand as a mathematical abstraction of a *computational effect* (in the sense of [26]) happening in conjunction with state transitions of the automaton; A is the set of inputs; and B is the set of outputs. According to this view, e.g. nondeterminism is the underlying effect of nondeterministic finite state machines. Analogously, we show that certain (nondeterministic) transformations of the pushdown store form the underlying effect of pushdown automata, etc. By

instantiating the operational analysis of computational effects from [28] to our setting we arrive at syntactic fixpoint expressions representing $\mathbb{T}$-automata and prove a Kleene-style theorem for them, thus generalizing previous work of the third author [38].

A crucial ingredient of our framework is the *generalized powerset construction* [39], which serves as a coalgebraic counterpart of classical Rabin-Scott determinization algorithm [30]. It allows us to define *trace semantics* of $\mathbb{T}$-automata and fixpoint expressions denoting their behavior.

We give a formal argument indicating that it is unlikely to capture languages beyond $\mathsf{NTIME}(n)$ using coalgebraic (trace) semantics in a straightforward way (i.e., in our case, using the generalized powerset construction) — the phenomenon known before as a property of *real-time machines* [4]. The requirement to be real-time is an inherent coalgebraic phenomenon of reactivity (or productivity), which restricts the class of behaviors that can be modeled. This led us to formulate a more general *observational semantics*, that allows us to take into account internal (or silent τ-)transitions coalgebraically. The latter furthermore enabled us to capture recursively enumerable languages by a special instance of $\mathbb{T}$-automata called tape automata and that are very similar to Turing machines. Capturing any kind of Turing complete formalism by coalgebras has been a long standing open problem, to which the present paper provides an answer. This results brings us closer to having a coalgebraic Chomsky hierarchy and a new abstract understanding of computability theory.

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 addressed rational models (e.g. rational streams, regular languages) from which we note [35] (regular languages and finite automata), [15] (bialgebraic view of Kleene algebra and regular expressions), [38, 25, 27, 3] (coalgebraic regular expressions). More recently, some further generalizations were proposed. In recent work [43] a coalgebraic model of context-free grammars is given, without however an analogous treatment of push-down automata. In [13] some initial results on $\mathbb{T}$-automata over stacks by the first author were presented, which the present work extends considerably.

2 Preliminaries: Deterministic Moore Automata, Coalgebraicaly

In this section we recall the main definitions and existing results on coalgebraic modelling of state machines. 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. The central notion in this paper is that of an *F-coalgebra* is a pair $(X, f : X \to FX)$ where F is an endofunctor on $\mathbf{Set}$ called *transition type*, X is a set called the *state space* and f is a map called *transition structure*. We shall occasionally 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: $h\colon X \to Y$ is a coalgebra morphism from $(X, f\colon X \to FX)$ to $(Y, g\colon Y \to FY)$ iff $g \circ h = Fh \circ f$. 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 $\hat{f}\colon X \to \nu F$ for the unique homomorphism from (X, f) to $(\nu F, \iota)$.

Our core example is the standard formalization of Moore automata as coalgebras [34]. For the rest of the paper we fix a finite set A of *actions* and a set B of *outputs*. We call the functor $L = B \times (\text{–})^A$ 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 $B = 2$) 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: $\partial_\epsilon(x) = x$; $\partial_{aw}(x) = \partial_a(\partial_w(x))$.

The final L-coalgebra νL always exists and is carried by the set of all *formal power series* B^{A^*}. The transition structure is given by $o(\sigma) = \sigma(\epsilon)$ and $\partial_a(\sigma) = \lambda w.\sigma(aw)$ for every formal power series $\sigma : A^* \to B$. The unique homorphism from an L-coalgebra X to the B^{A^*} assigns to every state $x_0 \in X$ a formal power series that we regard as the *(trace) 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 formal languages on A and the trace semantics assigns to every state of a given finite deterministic automaton the language accepted by that state.

Definition 2.1 (Trace semantics, Trace equivalence). Given an L-coalgebra (X, f) and $x \in X$, we write $[\![-]\!]_X : X \to B^{A^*}$ for the unique L-coalgebra morphism. For every $x \in X$ we call $[\![x]\!]_X$ the *trace semantics* of x (w.r.t. X). *Trace equivalence* identifies exactly those x and y for which $[\![x]\!]_X = [\![y]\!]_Y$ (for possibly distinct coalgebras X and Y); this is denoted by $x \sim y$.

The following result easily follows by definition (see e.g. [35, Theorem 9.1]).

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

It is well-known that Moore automata, i.e. *finite* L-coalgebras, can be characterized in terms of the formal power series occurring as their trace semantics (see e.g. [35]).

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 Moore automata (see e.g. Eilenberg [9]).

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 $B = k$ is a semiring, in which case one usually also speaks of *rational formal power series* as behaviours of finite weighted automata over k (see e.g. [8]). 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' [12, 7]. Regular formal power series as the semantics of precisely the finite L-coalgebras are a special instance of a general coalgebraic phenomenon [2, 25]. 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 the final morphism $\hat{f} : X \to \nu F$. Then ϱF is a subcoalgebra of νF with an isomorphic transition structure map 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 [2] for more 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 the solution of a system of recursive equations of the form

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

which should be read as follows: for all $1 \leq i, j \leq k$, $\partial_{a_j}(\sigma_i) = \sigma_{i_j}$ 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 (2.1) can be compactly written as

$$\sigma_i = \mu\sigma_i.\ (a_1.\sigma_{i_1} \pitchfork \cdots \pitchfork a_n.\sigma_{i_n} \pitchfork c_i) \tag{2.2}$$

where μ is the fixpoint operator binding the occurrences of σ_i in the right term. One can successively eliminate all the σ_i except σ using the equations (2.2) as assignments and thus obtain a "solution" $\sigma = t$ of (2.1) in σ where t is a closed term given by the following grammar:

$$\gamma ::= \mu X.\ (a_1.\delta \pitchfork \cdots \pitchfork a_n.\delta \pitchfork B) \qquad\qquad \delta ::= X \mid \gamma \tag{2.3}$$

Here X refers to an infinite stock of variables. Equation $\sigma = t$ is then nothing but a condensed representation of system (2.1) and as such it uniquely defines σ. On the other hand, expressions of the form (2.3) suggest a far reaching generalization of classical regular expressions and the fact that they capture exactly regular power series together with Proposition 2.4 can be viewed as a coalgebraic reformulation of Kleene's theorem. This view has been advanced recently (in a rather more general form) in [38, 27] and is of crucial importance for the present work.

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 the classical Brzozowski's theorem for regular expressions [5]. Given an expression $e = \mu x.\ (a_1.e_1 \pitchfork \cdots a_n.e_n \pitchfork c)$, 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 (2.4).*

3 Monads and Algebraic Theories

In the previous section we have presented a coalgebraic picture of deterministic Moore automata, essentially capturing the Type-3 level of Chomsky hierarchy (modulo the generalization from languages to power series). In order to deal with other levels we introduce *(finitary) monads* and *algebraic theories* as a critical ingredient of our formalization, thus building on top of the recent previous work [18, 39].

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 $(T, \eta, -^\dagger)$ consists of an object assignment T sending sets to sets, a family of maps $\eta_X : X \to TX$ and an operator, called *Kleisli lifting*, sending any $f : X \to TY$ to $f^\dagger : TX \to TY$. These data are subject to the following axioms:

$$\eta^\dagger = \mathrm{id}, \qquad f^\dagger \eta = f, \qquad (f^\dagger g)^\dagger = f^\dagger g^\dagger.$$

In this work we find it easier to work with It is well-known that the definition of a monad as a Kleisli triple is equivalent to the usual definition of a monad $\mathbb{T}$ as an endofunctor T equipped with natural transformations $\eta : Id \to T$ (*unit*) and $\mu : T^2 \to T$ (*multiplication*). A $\mathbb{T}$-*algebra* is given by a set X and a map $f : TX \to X$ satisfying standard coherence conditions: $f\eta_X = \mathrm{id}_X$ and $\mu_X Tf = f\mu_X$, and a morphism of $\mathbb{T}$-algebras is just a morphism of algebras for the functor T (see [24]). The category of $\mathbb{T}$-algebras and their morphisms is called *Eilenberg-Moore category of* $\mathbb{T}$ and is denoted by $\mathbf{Set}^{\mathbb{T}}$.

In what follows we occasionally use Haskell-style do-notation: for any $p \in TX$ and $q : X \to TY$ we write $\mathsf{do}\ x \leftarrow p; q(x)$ to denote $q^\dagger(p) \in TY$; and $p \in T(X \times Y)$ we write $\mathsf{do}\ \langle x, y\rangle \leftarrow p; q(x, y)$. This notation allows for a more convenient point-full reasoning with Kleisli morphisms, effectively avoiding potential tedious calculations due to strength. A monad $\mathbb{T}$ 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 an equivalent presentation in terms of (finitary) *algebraic theories* as we now outline.

Definition 3.2 (Algebraic 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*. Terms over Σ are constructed from operations and variables in the usual way. An *algebraic theory* over Σ is given by a set of term equations closed under under inference of the standard equational logic. As usual, an algebraic theory arises as the deductive closure of a set of its *axioms*.

Given an algebraic theory $\mathcal{E}$ over Σ we can form a monad $\mathbb{T}$ as follows: TX is the equivalence class of terms of the theory over free variables from X; $\eta_X : X \to TX$ casts a variable to a term; given $\sigma : X \to TY$ and $p \in TX$, $\sigma^\dagger(p)$ is the variable substitution $p[x \mapsto \sigma(x)]$. Conversely, we can pass from $\mathbb{T}$ to an algebraic theory $\mathcal{E}$ by

introducing an operation $f_a : n \to 1$ for each element a of Tn. Such an operation can be interpreted as a map $[\![f_a]\!] : TX^n \to TX$ by putting $[\![f_a]\!](t_1, \ldots, t_n) = (\lambda i.\, t_i)^\dagger(a)$. This yields a semantics of algebraic terms over TX and we take as the algebraic theory in question the class of all term equations valid over any TX. The above conversions between finitary monads and algebraic theories are mutually inverse. In the sequel we denote by $\mathcal{E}_\mathbb{T}$ the algebraic theory corresponding to $\mathbb{T}$. Notably, $\mathbb{T}$-algebras are then exactly the models of the algebraic theory $\mathcal{E}_\mathbb{T}$.

Example 3.3 (Monads, Algebraic theories). Standard examples of computationally relevant monads include (cf. [26]):

– The *finite and unbounded powerset monads* $\mathcal{P}_\omega$ and $\mathcal{P}$. Only the first one is finitary and corresponds to the algebraic theory of join-semilattices with bottom.
– The *store monad* over a store S. The functorial part given as $X \mapsto (X \times S)^S$. 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. As shown in [29], if V is finite then the corresponding store monad can be captured by an algebraic theory over operations $\{lookup_l : V \to 1\}_{l \in L}$ and $\{update_{l,v} : 1 \to 1\}_{l \in L, v \in V}$.
– The *continuation monad*. Given any set R, the assignment $X \mapsto (R^X \to R)$ yields a monad under the following definitions:

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

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

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

Example 3.4 (Semimodule monad, Semimodule theory). Given a semiring R, the semimodule monad $\mathbb{T}_R$ 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

$$r_1 \cdot x_1 + \cdots + r_n \cdot x_n \qquad (r_i \in R, x_i \in X). \tag{3.1}$$

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 $\sigma^\dagger(p)$ applies the substitution $\sigma : X \to \langle Y \rangle_R$ to $p \in \langle X \rangle_R$ and renormalizes the result as expected.

The semimodule monad corresponds to the algebraic theory of R-semimodules. Explicitly, we have a constant $\emptyset : 0 \to 1$, a binary operation $+ : 2 \to 1$, and every $r \in R$ gives rise to a unary operation $\bar{r} : 1 \to 1$. Terms of the theory are then build over these operations and modulo the laws of commutative monoids for $+$ and $\emptyset$, plus the following ones of a (left) action of R on a monoid:

$$\begin{aligned} \bar{r}(x + y) &= \bar{r}(x) + \bar{r}(y) & \bar{r}(\emptyset) &= \emptyset \\ \bar{r}(x) + \bar{s}(x) &= \overline{r + s}(x) & \bar{0}(x) &= \emptyset \\ \bar{r}(\bar{s}(x)) &= \overline{r \cdot s}(x) & \bar{1}(x) &= x \end{aligned}$$

It can be shown that any term can by normalized to the form $\bar{r}_1(x_1) + \cdots + \bar{r}_n(x_n)$ and the latter coherently represents the element (3.1) of $\langle X \rangle_R$, which allows us to identify them. Some notable instances of the semimodule monad $\mathbb{T}_R$ for semirings R of interest are the following:

– If R is the Boolean semiring $\{0, 1\}$ then $\mathbb{T}_R$ is (isomorphic to) the finite powerset monad $\mathcal{P}_\omega$.
– If R is the semiring of natural numbers then $\mathbb{T}_R$ 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 $\mathbb{T}_R$ is the monad of *finite valuations* used for modelling probabilistic computations [42]. Two other well-known monads of *finite distributions* and *finite subdistributions* serving the same purpose embed into $\mathbb{T}_R$: the formal sums (3.1) for them are requested to satisfy the additional constraints $r_1 + \cdots + r_n = 1$ and $r_1 + \cdots + r_n \leq 1$, respectively.

Finally, the following example is critical for modelling the push-down store.

Example 3.5 (Stack monad, Stack theory). Given a finite set of stack symbols Γ, the *stack monad (over Γ)* is the submonad $\mathbb{T}$ 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 $u \in \Gamma^*$,

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

Intuitively, a function $f : X \to TY$ (cf. Example 3.3) 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 theory* 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 variables under an operation represent continuations, i.e. computations that will be performed once the operation has completed its task, cf. [28]):

– $pop(x_1, \ldots, x_n, y)$ proceeds with y if the stack is empty; otherwise it removes the top element of 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.

These operations are subject to the following axioms:

$$\begin{aligned} push_i(pop(x_1, \ldots, x_n, y)) &= x_i \\ pop(push_1(x), \ldots, push_n(x), x) &= x \\ pop(x_1, \ldots, x_n, pop(y_1, \ldots, y_n, z)) &= pop(x_1, \ldots, x_n, z) \end{aligned}$$

As shown in [13] the stack theory is precisely the algebraic theory of the stack monad.

Finally we introduce a monad and the corresponding theory underlying the tape of a Turing machine. We introduce the following notation: given an integer $i \in \mathbb{Z}$, a nonnegative integer k and a map $\sigma : \mathbb{Z} \to \Gamma$, we write $\sigma =_{i \pm k} \sigma'$ ($\sigma =^{i \pm k} \sigma'$) if $\sigma(j) = \sigma'(j)$ for all j such that $|i - j| \leq k$ ($|i - j| > k$).

Definition 3.6 (Tape monad, Tape theory). Let Γ be a finite set of tape symbols. The *tape monad (over Γ)* is the submonad $\mathbb{T}$ of the store monad $(- \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})^{\mathbb{Z} \times \Gamma^{\mathbb{Z}}}$ for which TX consists of exactly those maps $\langle r, z, t\rangle : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to (X \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})$, which satisfy restriction: there is $k \geq 0$ such that for any $i, j \in \mathbb{Z}$ and $\sigma, \sigma' : \mathbb{Z} \to \Gamma$ if $\sigma =_{i \pm k} \sigma'$ then

$$t(i,\sigma) =_{i\pm k} t(i,\sigma'), \qquad r(i,\sigma) = r(i,\sigma'), \qquad |z(i,\sigma) - i| \leq k,$$
$$t(i,\sigma) =^{i\pm k} \sigma, \qquad z(i,\sigma) = z(i,\sigma'), \qquad t(i,\sigma_{+j}) = t(i+j,\sigma)_{+j},$$
$$r(i,\sigma_{+j}) = r(i+j,\sigma), \qquad z(i,\sigma_{+j}) = z(i+j,\sigma) - j.$$

where σ_{+j} denotes $\sigma \circ (\lambda i.\, i + j)$. The *tape signature* w.r.t. $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ consists of the operations $read : n \to 1$, $write_i : n \to 1$ $(1 \leq i \leq n)$, $lmove : 1 \to 1$, $rmove : 1 \to 1$, which we interpret over any TX as follows:

$$[\![read]\!](p_1, \ldots, p_n)(z,\sigma) = p_{\sigma(z)}(z,\sigma) \qquad [\![lmove]\!](p)(z,\sigma) = p(z-1,\sigma)$$
$$[\![write_i]\!](p)(z,\sigma) = p(z,\sigma[z \mapsto \gamma_i]) \qquad [\![rmove]\!](p)(z,\sigma) = p(z+1,\sigma)$$

where $\sigma[z \mapsto \gamma]$ overwrites σ with the assignment $z \mapsto \gamma$, that is: $\sigma[z \mapsto \gamma](z) = \gamma$ and $\sigma[z \mapsto \gamma](z') = \sigma(z')$ for $z' \neq z$. The *tape theory* w.r.t. Γ consist of all those equations $p = q$ in the tape signature, which are valid over every TX.

In contrast to the stack theory the tape theory is given indirectly. We leave the question of finding an appropriate complete axiomatization of the tape theory for future work. Meanwhile, we report the following surprising result:

Theorem 3.7. *The tape theory is not finitely axiomatizable.*

Proof (Sketch). It is easy to verify that for any i, j, k the following identity

$$write_i(lmove^k(write_j(rmove^k(x)))) = lmove^k(write_j(rmove^k(write_i(x))))$$

belongs to the tape theory. The left-hand term represents a computation that first writes γ_i at the current head position then moves k steps to the left, writes γ_j and finally moves k steps back to the right; the left-hand computation does the same in a different order. It is then possible for any finite set $\mathcal{A}$ of identities to construct a model that does not satisfy the above identity for a suitable k depending on the size of $\mathcal{A}$. □

4 Reactive $\mathbb{T}$-algebras and $\mathbb{T}$-automata

As in Section 2 we consider a finite set of actions A. We first consider $\mathbb{T}$-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 $\mathbb{T}$-algebra with dynamic behaviour (making it into a coalgebra) and hence we call such structures *reactive* $\mathbb{T}$-algebras.

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

Remark 4.2. The definition of a reactive $\mathbb{T}$-algebra is an instance of a more general construction stemming from the seminal work of Turi and Plotkin [41]. Any endofunctor $F : \mathbf{Set} \to \mathbf{Set}$ equipped with a distributive law $\delta : \mathbb{T}F \to F\mathbb{T}$ is known to lift to the Eilenberg-Moore category $\mathbf{Set}^{\mathbb{T}}$. Under $F = L$ there is a standard distributive law, given by

$$\delta(p) = \langle f(\mathsf{do}\,\langle b, c\rangle \leftarrow p; \eta(b)), \lambda a.\ \mathsf{do}\,\langle b, c\rangle \leftarrow p; c(a)\rangle$$

where $f : TB \to B$ is the $\mathbb{T}$-algebra structure on B. Then a reactive $\mathbb{T}$-algebra is simply a coalgebra in $\mathbf{Set}^{\mathbb{T}}$ for the lifting of L. Putting it yet differently, a reactive $\mathbb{T}$-algebra is a δ-*bialgebra* for the given δ [15] (see also [21]).

Given a $\mathbb{T}$-algebra B, the set of all formal power series B^{A^*} being the final L-coalgebra can be viewed as a reactive $\mathbb{T}$-algebra with the pointwise $\mathbb{T}$-algebra structure, for which ∂_a and o are easily seen to be $\mathbb{T}$-algebra morphisms. Since any reactive $\mathbb{T}$-algebra is an L-coalgebra, reactive $\mathbb{T}$-algebras inherit the general coalgebraic theory from Section 2. In particular, we use for reactive $\mathbb{T}$-algebras the same notions of trace semantics and trace equivalence as for L-coalgebras.

Definition 4.3 ($\mathbb{T}$-automaton). Suppose, $\mathbb{T}$ 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 $\mathbb{T}$-algebra morphism. A $\mathbb{T}$-*automaton* m is given by a triple of maps

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

where a^m is a $\mathbb{T}$-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 $\mathbb{T}$ in ($\bigstar$) to be finitary for the sake of computational feasibility of $\mathbb{T}$-automata and to be able to represent them using finite syntax. For technical reasons, it is sometimes convenient to drop this restriction (e.g. in Section 8 where $\mathbb{T}$ is the continuation monad). This is not in conflict with Definition 4.3, for since we apply $\mathbb{T}$ to finite sets only it can be used interchangeably with its finitary coreflection $\mathbb{T}_\omega$ whose object part is defined by $T_\omega X = \bigcup_{Y \subseteq X, |Y| < \omega} TY$.

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

In order to introduce the trace semantics of a $\mathbb{T}$-automaton it suffices to convert it into a reactive $\mathbb{T}$-algebra, for the trace semantics of the latter is settled by Definition 2.1. This conversion is called the *generalized powerset construction* [30] as it generalizes the classical Rabin and Scott NFSM determinization [39]. Observe that LTX is a $\mathbb{T}$-algebra, since TX is the free $\mathbb{T}$-algebra on X and L lifts to $\mathbf{Set}^{\mathbb{T}}$ (see Remark 4.2). Therefore, given a $\mathbb{T}$-automaton ($\bigstar$), $m : X \to B \times (TX)^A$ there exists a unique $\mathbb{T}$-algebra morphism $m^\sharp : TX \to B \times (TX)^A$ such that $m^\sharp \eta = m$. This $m^\sharp$ is a reactive $\mathbb{T}$-algebra on TX. Therefore, we define the trace semantics of ($\bigstar$) as follows:

$$[\![x]\!]_m = [\![\eta(x)]\!]_{TX}.$$

Remark 4.5. Due to the 1-1-correspondence of m and $m^\sharp$ given by freeness of TX, $\mathbb{T}$-automata bijectively correspond to reactive $\mathbb{T}$-algebras whose carrier is a free algebra on a finite set; but we find it useful to retain the distinction.

Note that the generalized powerset construction does not reduce a $\mathbb{T}$-automaton to a Moore automaton over TX as TX need not be finite, although when it is the case, e.g. $\mathbb{T} = \mathcal{P}_\omega$, the semantics of a $\mathbb{T}$-automaton falls within regular power series, which is precisely the reason why the languages recognized by deterministic and nondeterministic FSM coincide. Surprisingly, all $\mathbb{T}$-automata with a finite B have the same property, which is a corollary of Theorem 8.1 we prove in Section 8.

Proposition 4.6. *For every $\mathbb{T}$-automaton ($\bigstar$) with finite B and $x \in X$, $[\![x]\!]_m : A^* \to B$ is regular.*

We are now ready to introduce fixpoint expressions for $\mathbb{T}$-automata similar to (2.3).

Definition 4.7 (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:

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

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 consider terms modulo α-conversion.

Let $\mathbb{T}$ be a finitary monad, corresponding to an algebraic theory $\mathcal{E}$ over the signature Σ and let B be a finitely generated $\mathbb{T}$-algebra over a finite set of generators B_0. Let us denote by E_{Σ,B_0} the set of all reactive expressions over Σ and B_0. We define a reactive $\mathbb{T}$-coalgebra structure on E_{Σ,B_0}. First, notice that E_{Σ,B_0} is obviously a Σ-algebra. Then we introduce the L-coalgebra structure on E_{Σ,B_0} as follows: first notice that expressions b according to the β-clause in Definition 4.7 are just Σ-terms on the generators from B_0; for every Σ-term t in n variables let $t^B : B^n \to B$ denote the map evaluating t in B and define

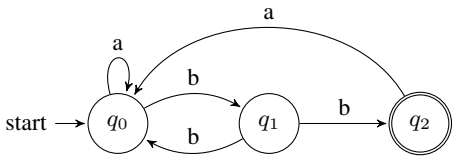
$$\begin{aligned} o(f(e_1, \cdots, e_n)) &= f^B(o(e_1), \dots, o(e_n)), \\ \partial_{a_i}(f(e_1, \cdots, e_n)) &= f(\partial_{a_i}(e_1), \dots, \partial_{a_i}(e_n)), \\ o(\mu x.\ (a_1.e_1 \pitchfork \cdots \pitchfork a_n.e_n \pitchfork b)) &= t^B(b_1, \dots, b_k), \\ \partial_{a_i}(\mu x.\ (a_1.e_1 \pitchfork \cdots \pitchfork a_n.e_n \pitchfork b)) &= e_i[\mu x.\ (a_1.e_1 \pitchfork \cdots \pitchfork a_n.e_n \pitchfork b)/x], \end{aligned}$$

where $b = t(b_1, \dots, b_k)$ with $b_1, \dots, b_k \in B_0$.

We call on Definition 2.1 to endow E_{Σ,B_0} with the trace semantics $[\![-]\!] : \mathsf{E}_{\Sigma,B_0} \to B^{A^*}$ and with the trace equivalence relation $\sim$.

Theorem 4.8. *The quotient $\mathsf{E}_{\Sigma,B_0}/\!\sim$ is a reactive $\mathbb{T}$-algebra whose L-coalgebra part is inherited from E_{Σ,B_0} and whose $\mathbb{T}$-algebra part is a quotient of the Σ-algebra structure on E_{Σ,B_0}.*

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



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

Fig. 1. A $\mathcal{P}_\omega$-automaton over $A = \{a, b\}$, $B = \{\top, \bot\}$ as a system of recursive definitions (left); as a nondeterministic FSM (right); as a reactive expression (bottom).

The following theorem is the main result of this section — a Kleene type theorem. Like its classical counterpart it allows to convert $\mathbb{T}$-automata to closed expressions and vice versa.

Theorem 4.9 (Kleene theorem). *For any expression $e \in \mathsf{E}_{\Sigma,B_0}$ there is a corresponding $\mathbb{T}$-automaton (★) and $x \in X$ such that $[\![e]\!] = [\![x]\!]_m$; and conversely for any $\mathbb{T}$-automaton (★) and $x \in X$ there is an expression $e \in \mathsf{E}_{\Sigma,B_0}$ such that $[\![e]\!] = [\![x]\!]_m$.*

Fig. 1 depicts a simple instance of the general correspondence established by Theorem 4.9 in the particular standard case of nondeterministic FSM.

5 $\mathbb{T}$-automata: Examples

As indicated above, a nondeterministic FSM is a specific case of a $\mathbb{T}$-automaton under $B = 2$ and $\mathbb{T} = \mathcal{P}_\omega$. More generally, we have the following definition.

Definition 5.1 (Weighted $\mathbb{T}$-automata). *Weighted $\mathbb{T}$-automaton* is a $\mathbb{T}$-automaton (★) with $\mathbb{T}$ being the semimodule monad for the semiring R (see Example 3.4).

Let R be the underlying semiring of a semimodule monad $\mathbb{T}$. Besides $R = B = 2$ in which case we obtain nondeterministic FSMs, we obtain standard weighted automata [8] under $R = B = N$ (B is the free $\mathbb{T}$-algebra finitely generated by $\{1\}$).

Weighted $\mathbb{T}$-automata can be further generalized as follows. We call a monad *additive* (cf. [6]) if the corresponding algebraic theory supports operations $+ : 2 \to 1$ and $\emptyset : 0 \to 1$ subject to the axioms of commutative monoids. We call a $\mathbb{T}$-automaton *additive* if $\mathbb{T}$ is additive. Additive automata allow for a more relaxed syntax of reactive expressions. Specifically, we define *additive reactive expressions* as closed *guarded* expressions over an additive theory given by the grammar

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

where guardedness means that for any subterm of the form $\mu x.\,e$ (the recursive call of) x is guarded in e, which is defined by induction over e as follows: x is guarded in b, in any variable $x' \neq x$, in any $\mu x.\,e'$ and in any $a.e'$; and x is guarded in $f(e_1, \ldots, e_n)$ whenever x is guarded in each of the e_i.

Given a reactive expression we obtain an additive reactive expression by replacing recursively each $\pitchfork$ with $+$. Conversely, any additive reactive expression can be transformed to a reactive one. The latter transformation is inverse to the former modulo $\sim$.

Example 5.2 (Rabin's probabilistic automata). Rabin's probabilistic automata [31] can be modelled as weighted $\mathbb{T}$-automata over the semiring $[0, \infty)$ under the standard arithmetic operations. As we mentioned in Example 3.4), finitary probability distributions form a monad, which allows for a slightly more elegant formalization: we take the finitary distribution monad as $\mathbb{T}$ and $B = [0, 1]$. The algebraic theory of $\mathbb{T}$ defined over the signature of binary operations $+_p : 2 \to 1$ with $p \in [0, 1]$. The expression $x +_p y$ denotes probabilistic choice between x and y: the left branch is chosen with probability p and the right branch is chosen with probability $1 - p$. The axioms of this theory can be found e.g. in [17].

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

Example 5.3 (Segala $\mathbb{T}$-automata). *(Simple) Segala systems* [36, 37] 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 a probability distribution functor. Although $\mathcal{P}\mathcal{D}$ is not a monad, as elaborated in [16], it can be modelled by a monad $\mathbb{T}$ whose functorial part is the composition CM of two functors given as follows: for any X, MX are finite valuations over X (see Example 3.4); for any semimodule U, $C(U)$ consists of all subsets of U, which are *convex* and nonempty. Convexity of a set S here means that a convex combination $p_1 \cdot \xi_1 + \cdots + p_n \cdot \xi_n$, i.e. $\sum_i p_i = 1$, belongs to S once $\xi_i \in S$ for any i. *Segala $\mathbb{T}$-automata* generalize non-deterministic automata by replacing the powerset functor $\mathcal{P}$ with CM. Concretely, in the generic definition (★) we take $B = 2$ and $\mathbb{T}$ defined as above.

A radically different kind of examples is offered by submonads of the store monad. A prominent instance of such is the stack monad (Example 3.5), which we use for modelling push-down automata.

Definition 5.4 (Stack $\mathbb{T}$-automaton). *Stack $\mathbb{T}$-automaton* is a $\mathbb{T}$-automaton (★) for which

- $\mathbb{T}$ 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 is k such that $p(wu) = p(w)$ whenever $|w| \geq k$;
- $a^{\mathsf{m}} : TB \to B$ 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: $a^{\mathsf{m}}(r, t)(s) = r(s)(t(s))$.

Intuitively, $o^{\mathsf{m}} : X \to B \subseteq 2^{\Gamma^*}$ models the acceptance condition by final states and the stack content. As B obeys essentially the same constraints as TX, scanning an unbounded portion of the stack by o^{m} is disallowed; the role of the algebraic structure a^{m} is roughly to trace acceptance conditions back along the transition structure t^{m}.

In terms of algebraic theories, B is finitely generated over the set of generators $\{0,1\}$ and as such is a quotient of $T2$ under additional laws: $push_i(0) = 0$ and $push_i(1) = 1$.

Theorem 5.5. *Let m be a stack $\mathbb{T}$-automaton. Then for any $x_0 \in X$ and any $\gamma_0 \in \Gamma$,*

$$\left\{w \in A^* \mid [\![x_0]\!]_m(w)(\gamma_0)\right\} \tag{5.2}$$

is a real-time deterministic context-free language. The converse is also true: for any real-time deterministic context-free language $\mathcal{L} \subseteq A^*$ there exists a stack $\mathbb{T}$-automaton (★) such that $\mathcal{L}$ can be represented as (5.2) with some $x_0 \in X$, $\gamma_0 \in \Gamma$.

As we shall see later, it is not difficult to obtain an analogous characterization of context-free languages for which the "real-time" clause is dropped (essentially because for push-down automata the restriction of being real-time is vacuous). However, as we shall see in the next section (Theorem 6.5), this restriction, being somewhat inherent for coalgebraic models, presents an actual boundary for capturing by $\mathbb{T}$-automata formal languages beyond the context-free ones.

6 Monad Tensors for Store and Nondeterminism

Tensor products of monads (resp. algebraic theories) have been introduced by Freyd [11] in the context of universal algebra. Later, computational relevance of this operation has been demonstrated by Hyland et al. [14]. Here, we use tensors of monads as a tool for studying $\mathbb{T}$-automata.

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 any $f : n \to 1$ of $\mathcal{E}_1$ and any $g : m \to 1$ 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 $\mathbb{T}_1$ and $\mathbb{T}_2$, their tensor product arises from the algebraic theory $\mathcal{E}_{\mathbb{T}_1} \otimes \mathcal{E}_{\mathbb{T}_2}$.

Intuitively, 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* and (2) tensors with *semimodule monads*.

It has been shown in [14] that tensoring with the store monad is equivalent to the application of the store monad transformer sending any monad $\mathbb{T}$ to the *store monad transform* $\mathbb{T}_S$ whose functorial part is given by $T_S X = T(X \times S)^S$. Here we establish a similar result about the stack monad (Example 3.5).

Proposition 6.2. *Let $\mathbb{S}$ be the stack monad over Γ. Then for any finitary $\mathbb{T}$, $\mathbb{S} \otimes \mathbb{T}$ is the submonad $\mathbb{R}$ of the store monad transform of $\mathbb{T}$ with Γ^* as the store, for which*

– $p : \Gamma^* \to T(X \times \Gamma^*)$ *is in* $\mathbb{R}X$ *iff there exists* m *such that* $p(su) = \mathsf{do}\,\langle x, s'\rangle \leftarrow p(s); \eta\langle x, s'u\rangle$ *whenever* $|s| \geq m$*;*
– *the monad structure is inherited from the monad transform.*

One can thus combine two stacks by computing the tensor square of the stack monad. Specifically, the resulting monad $\mathbb{T}$ has the following maps as inhabitants of TX:

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

This allows one to define $\mathbb{T}$-stack automata over two and more stacks analogously to the one-stack case from Definition 5.4. Before we do this formally in Definition 6.4 we discuss the perspectives of forming tensors with semimodule monads.

Proposition 6.3 (Freyd [11]). *Tensor product of any finitary monad with a semimodule monad is isomorphic to some semimodule monad.*

Proposition 6.3 in conjunction with Proposition 6.2 offer two perspectives on machines with memory and nondeterminism. E.g. we shall consider the tensor product of $\mathcal{P}_\omega$ with the stack monad to model 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 X-values weighted in the semiring of transition relations $\mathcal{P}(\Gamma^* \times \Gamma^*)$.

Definition 6.4 (Multi-stack nondeterministic $\mathbb{T}$-automaton). A *Multi-stack nondeterministic* $\mathbb{T}$*-automaton* is a $\mathbb{T}$-automaton (★) for which

– $\mathbb{T}$ is the tensor of m copies of the stack monad and $\mathcal{P}_\omega$;
– 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 if for any i, $|w_i| \geq k$ then $p(w_1u_1, \ldots, w_mu_m) = p(w_1 \ldots, w_m)$;
– for any $s \in (\Gamma^*)^m$, $f : (\Gamma^*)^m \to \mathcal{P}_\omega(B \times (\Gamma^*)^m) \in TB$

$$a^{\mathfrak{m}}(f)(s) \quad \text{iff} \quad \exists s' \in (\Gamma^*)^m.\, \exists p \in B.\ f(s)(p, s') \wedge p(s').$$

We now obtain the following result.

Theorem 6.5. *For any* m *let* $\mathcal{L}_m$ *be the following class of all languages* $\{w \in A^* \mid [\![x_0]\!]_{\mathfrak{m}}(w)(\gamma_0, \ldots, \gamma_0)\}$ *with* $\mathfrak{m}$ *ranging over nondeterministic multistack* $\mathbb{T}$*-automata with* m *stacks,* x_0 *ranging over the state space of* $\mathfrak{m}$ *and* γ_0 *ranging over* Γ*. Then*

– $\mathcal{L}_1$ *contains exactly context-free languages;*
– *for all* $m > 2$*,* $\mathcal{L}_m$ *contains exactly nondeterministic linear time languages, i.e.* $\mathcal{L}_m = \mathsf{NTIME}(n)$*;*
– $\mathcal{L}_2$ *sits properly between* $\mathcal{L}_1$ *and* $\mathcal{L}_3$*.*

Theorem 6.5 shows, on the one hand, that the coalgebraic formalization of nondeterministic pushdown automata as nondeterministic $\mathbb{T}$-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 $\mathbb{T}$-automata. This is not surprising in view of the early work on (quasi-)real-time recognizable languages [4], which underlies the proof of Theorem 6.5. 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 an easy corollary of Theorem 6.5 and Proposition 2.2 contrasting the results in [25, 3].

Corollary 6.6. *Trace equivalence of $\mathbb{T}$-automata is Π_1^0-complete.*

7 Context-free Languages and Algebraic Expressions

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

By Proposition 6.3, a nondeterministic $\mathbb{T}$-automaton over one stack is a specific case of a weighted $\mathbb{T}$-automaton (Definition 5.1). In this form it is rather similar to another example of a machine previously studied in the literature (e.g. [32, 19]) and which we also can formalize as a $\mathbb{T}$-automaton. We present the corresponding algebraic theories side by side.

Example 7.1 (Nondeterministic stack theory). The *nondeterministic stack theory* is obtained by tensoring $\mathcal{E}_{\mathcal{P}_\omega}$ with the stack theory. The result is a semimodule theory generated by the scalars o_i, u_i and e such that $pop_i = \overline{o_i}$, $push_i = \overline{u_i}$ and $e = \overline{empty}$ (in the notation of Example 3.4). Here the unary operations pop_i and $empty$ determine the decomposition of pop

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

in accordance with the tensor law and are defined as follows: $pop_i(x) = pop(\emptyset, \dots, x, \dots, \emptyset, \emptyset)$ (x is on the i-the position) and $empty(x) = pop(\emptyset, \dots, \emptyset, x)$.

Example 7.2 (Nondeterministic polycyclic theory). The *nondeterministic polycyclic theory* is obtained by tensoring $\mathcal{E}_{\mathcal{P}_\omega}$ with the theory of polycyclic monoids. Equivalently (see [14]), the nondeterministic polycyclic theory is the theory of the monad $\mathcal{P}_\omega(- \times M)$ where M is a *polycyclic monoid* [23]. The latter means that M is the monoid over a set of generators $0, g_1, \dots, g_k, \dots g_1^{-1}, \dots, g_k^{-1}$ satisfying identities

$$0g_i = g_i 0 = 0, \qquad g_i g_i^{-1} = 1, \qquad g_i g_j^{-1} = 0 \quad (i \neq j).$$

The number k is called the *rank* of M.

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 $\emptyset$ of the semiring to model failure in computing the right inverse, while the nondeterministic polycyclic theory has its own zero 0, which coexists with $\emptyset$. On the other hand, emptiness detection is natively available for stacks but not for monoids.

The common idea underlying monoid-based examples including Example 7.2 is to use the monoid structure to model various kinds of stores (stack(s), counter(s), etc.). The corresponding automata are called *valence automata* [32].

It is well-known that valence automata over polycyclic monoids of rank at least 2 recognize context-free languages. We would like to give a generic proof of this fact applying both to Example 7.1 and to Example 7.2.

First, observe that if R is idempotent then B can be partially ordered by putting $b \leq c$ iff $b + c = c$. Given a $\mathbb{T}_R$-automaton $\mathcal{m}$, and an initial state $x_0 \in X$ we can define the language *recognized by* $b \in B$

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

E.g. for the stack $\mathbb{T}$-automata we typically chose as $b \in 2^{\Gamma^*}$ the predicate distinguishing the initial stack configuration. For valence automata one standardly takes $B = M$ and $b = 1$.

Recall that the language of balanced parentheses, or *Dyck language* is a language $\mathcal{D}_n \subseteq \{(_1,)_1, \ldots, (_n,)_n\}^* = \mathcal{A}$ consisting of string of parentheses balanced in the expected way.

The following result is a consequence of the classical Chomsky-Schützenberger theorem.

Theorem 7.3. *Let α be a monoid morphism from $\mathcal{A}$ to the multiplicative structure of some idempotent R such that for some $b_0, b_1 \in B$, $\alpha(w) \cdot b_0 \geq b_1$ iff w is balanced. Suppose also, b_1 has the property that 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 $\mathcal{L}$ there is a $\mathbb{T}_R$-automaton recognizing $\mathcal{L}$ by b_1.*

It is easy to check the conditions of Theorem 7.3 for Examples 7.1 with $|\Gamma| > 1$ and 7.2 with $k > 1$. Contrasting [19] we cannot replace the polycyclic monoid in Example 7.2 by a free group and conclude by Theorem 7.3 that automata with free group memory recognize context-free languages. But as shown in [19] the latter is true once internal transitions are allowed.

We now explore how to directly incorporate context-free grammars in our framework as a generalized form of reactive expressions. Recall that for semimodule monads one can use the syntax of additive reactive expressions (5.1). Note that the corresponding algebraic signature contains w.l.o.g. besides the operations $\emptyset$ and $+$ only unary operations (see Proposition 6.3). Following the convention of Example 3.4, we can write those unary operations as scalar coefficients. Thus, the resulting grammar for additive reactive expressions takes the following form

$$\gamma ::= b \mid x \mid \emptyset \mid \mu x.\, \gamma \mid a.\gamma \mid r \cdot \gamma \mid \gamma + \gamma \qquad (a \in A, b \in B_0).$$

For example, for the nondeterministic stack theory this instantiates as follows:

$$\gamma ::= b \mid x \mid \emptyset \mid \mu x.\, \gamma \mid a.\gamma \mid o_i \cdot \gamma \mid u_i \cdot \gamma \mid e \cdot \gamma \mid \gamma + \gamma \qquad (a \in A, b \in B_0).$$

We generalize this further as follows.

Definition 7.4 (Algebraic expressions). *Algebraic expressions* w.r.t. B_0 and R_0 are guarded closed expressions given by the following grammar:

$$\gamma ::= b \mid x \mid \emptyset \mid \bullet \mid \mu x.\, \gamma \mid a.\gamma \mid r \cdot \gamma \mid \gamma + \gamma \mid \gamma \star \gamma$$

with $x \in X$, $a \in A$, $b \in B_0$ and $r \in R$ where guardedness means that for any subterm of the form $\mu x.\, e$, x is guarded in e and the latter is defined by induction in the same way as for additive expressions plus the new clause stating that x is guarded in $e \star e'$ whenever it is guarded in e.

We call an algebraic expression *primitive* if it is generated by the same grammar but with the clause $r \cdot \gamma$ replaced by the clauses $r \cdot \cdots \cdot r \cdot b$. The set of $\bullet$*-closed expressions* is defined inductively: all the clauses except $\bullet$ and $\gamma \star \gamma$ preserve $\bullet$-closedness and $e_1 \star e_2$ is closed when e_2 is closed.

The idea of the new operator $\star$ is to model sequential composition of two expressions; the role of $\bullet$ thereby is to mark the positions in the expression e_1 which become replaced by e_2 when the composition $e_1 \star e_2$ is calculated.

We turn the set of algebraic expressions into an L-coalgebra by defining the transition structure maps o and ∂_a as follows:

$$\begin{aligned} \partial_a(b) &= \emptyset & \partial_a(\emptyset) &= \emptyset & \partial_a(\bullet) &= \emptyset & \partial_a(r \cdot e) &= r \cdot \partial_a(r) \\ o(b) &= b & o(\emptyset) &= \emptyset & o(\bullet) &= \emptyset & o(r \cdot e) &= r \cdot o(e) \end{aligned}$$

$$\begin{aligned} \partial_a(\mu x.\, e) &= \partial_a(e[\mu x.\, e/x]) & \partial_a(e_1 + e_2) &= \partial_a(e_1) + \partial_a(e_2) \\ o(\mu x.\, e) &= o(e[\mu x.\, e/x]) & o(e_1 + e_2) &= o(e_1) + o(e_2) \end{aligned}$$

$$\partial_a(a.\, e) = e \qquad \partial_a(a'.\, e) = \emptyset \quad (a' \neq a) \qquad o(a.\, e) = \emptyset$$

$$\begin{aligned} \partial_a(e_1 \star e_2) &= \partial_a(e_1) \star e_2 + \bar{o}(e_1) \cdot \partial_a(e_2) \\ o(e_1 \star e_2) &= \bar{o}(e_1) \cdot o(e_2) + o(e_1) \end{aligned}$$

where $\bar{o}$ is defined analogously to o except for two cases $\bar{o}(\bullet) = 1$ and $\bar{o}(b) = \emptyset$. The fact that these definitions are well-founded easily follows from guardedness. The clauses for $e_1 \star e_2$ resemble behavioural differential equations for formal power series [35]. As usual, the given definitions induce the notion of trace equivalence $\sim$ on the set of all algebraic expressions, although we have to additionally assume $\bullet$-closedness to ensure that the induced trace semantics yields formal power series $A^* \to B$ (and not $A^* \to B \times R$).

Proposition 7.5. *The quotient of the set of* $\bullet$*-closed algebraic expressions over* B_0, R *forms a reactive* $\mathbb{T}_R$*-algebra.*

Algebraic expressions by definition generalize reactive expressions for weighted $\mathbb{T}$-automata, including the nondeterministic stack $\mathbb{T}$-automata. On the other hand, being more general than primitive algebraic expressions, they subsume context-free grammars, which are known to be representable by fixpoint expressions of a very similar form (this has been explored recently in a coalgebraic context in [43]).

8 CPS-transforms of $\mathbb{T}$-automata and r.e.-languages

Theorem 6.5 suggests that the present trace semantics is unlikely to produce languages beyond $\mathsf{NTIME}(n)$ under a computationally convincing choice of the components of (★). 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*, but use the formal power series $A^* \to B$ as the semantic domain instead of $A_\tau^* \to B$. The new *observational semantics* is supposed to be obtainable from the standard one by "eliminating" the unobservable action.

We argue briefly, why the general assumptions about the structure of a $\mathbb{T}$-automaton are not sufficient to define the observational semantics. Consider the Moore automaton $\mathcal{m} : X \to 2 \times X^{A_\tau}$ with $A = \{a\}$ and $B = \{b_0, b_1\}$. The underlying monad is the identity monad and $a^{\mathcal{m}}$ is the identity morphism. Let $X = \{x_0, x_1\}$, $o^{\mathcal{m}} = \{\langle x_0, b_0\rangle, \langle x_1, b_1\rangle\}$, $t^{\mathcal{m}} = \{\langle x_0, a, x_0\rangle, \langle x_0, \tau, x_1\rangle, \langle x_1, a, x_1\rangle, \langle x_1, \tau, x_1\rangle\}$. 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 $o^{\mathcal{m}}$, which cannot be determinized unless we presuppose a monoid structure on B. Before we make use of this observation, we introduce an important reformulation of our standard trace equivalence, which is of independent interest.

Let $a : TB \to B$ be a $\mathbb{T}$-algebra. We denote by $\mathbb{T}_B$ the continuation monad with $T_B X = B^X \to B$. We can map $\mathbb{T}$ to $\mathbb{T}_B$ by sending any $p : TX$ to $\kappa_X(p) = \lambda f.\, (a \cdot Tf(p)) \in T_B X$. This $\kappa : \mathbb{T} \to \mathbb{T}_B$ is a monad morphism; in fact, it is well known that for any monad $\mathbb{T}$ on a category with powers there is a bijective correspondence between Eilenberg-Moore algebras on B and monad morphisms from $\mathbb{T}$ to $\mathbb{T}_B$ (see e.g. Kock [22, Theorem 3.2]).

Now, given a $\mathbb{T}$-automaton (★), we define a $\mathbb{T}_B$-automaton[1] $\mathcal{m}_* : X \to B \times (T_B X)^A$ by $o^{\mathcal{m}_*} = o^{\mathcal{m}}$, $t^{\mathcal{m}_*} = \kappa_X t^{\mathcal{m}}$, $a^{\mathcal{m}_*} = \lambda t.\, t(\mathrm{id})$ where it is easy to see that $a^{\mathcal{m}_*} : T_B B \to B$ is a $\mathbb{T}$-algebra. We call this automaton the *CPS-transform*[2] of (★).

Theorem 8.1. *The trace semantics of a $\mathbb{T}$-automaton and of its CPS-transform agree; more precisely, for every $\mathbb{T}$-automaton (★) and state $x \in X$ we have: $[\![x]\!]_{\mathcal{m}} = [\![x]\!]_{\mathcal{m}_*}$.*

This theorem implies Proposition 4.6 announced previously in Section 4. Indeed, if B in (★) is finite then, by definition, $T_B X$ is also finite. Thus, the generalized powerset construction performed on the CPS-transform $\mathcal{m}_*$ yields a Moore automaton, and hence we obtain the desired result from Proposition 2.4.

We now proceed with the definition of the new semantics.

Definition 8.2 (ω-additive $\mathbb{T}$-automata). A $\mathbb{T}$-automaton (★) is *ω-additive* if B (besides being $\mathbb{T}$-algebra) is an algebra for the countably supported multiset monad.

In other words, for (★) to be ω-additive B needs to be a commutative monoid with infinite summation. 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 $\sum_i a_i$ the countable sum. It is easy to see that the ω-additive monoid structure extends from B to T_B pointwise.

[1] We abuse terminology here since $\mathbb{T}_B$ is not finitary (see Remark 4.4).

[2] CPS = *continuation-passing style*; we chose the name because the construction is reminiscent of tranforming a functional program into CPS.

Lemma 8.3. *If B is an ω-additive monoid and a $\mathbb{T}$-algebra then for any X, $T_B X$ is an ω-additive monoid.*

The ω-additive monoid structure on $T_B X$ allows us to define for any given $\mathbb{T}$-automaton over the alphabet A_τ a $\mathbb{T}_B$-automaton over A. To this end, we first form the CPS-transform of the given $\mathbb{T}$-automaton and then use infinite summation to get rid of unobservable actions τ: given a $\mathbb{T}$-automaton $m : X \to B \times (TX)^{A_\tau}$, we construct $m_v : X \to B \times (T_B X)^A$ with $a^{m_v} = a^{m_*} = \lambda t.\, t(\mathrm{id})$ and with t^{m_v}, o^{m_v} defined as

$$t^{m_v}(x_0, a) = \sum_{i=1}^{\infty} \mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0, \tau); \ldots; x_{i-1} \leftarrow t^{m_*}(x_{i-2}, \tau); t^{m_*}(x_{i-1}, a),$$

$$o^{m_v}(x_0) = o^{m_*}(x_0) + \sum_{i=1}^{\infty} \big(\mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0, \tau); \ldots; t^{m_*}(x_{i-1}, \tau)\big)(o^{m_*}).$$

We define the observational trace semantics for m to be the trace semantics for m_v.

Definition 8.4. Given a $\mathbb{T}$-automaton ($\bigstar$) over input alphabet A_τ, let $[\![x]\!]^\tau_m = [\![x]\!]_{m_v}$.

We proceed to define the class of $\mathbb{T}$-automata corresponding to classical Turing machines, for which the introduced observational trace semantics yields precisely all recursively enumerable languages.

Definition 8.5 (Tape $\mathbb{T}$-automaton). A *tape automaton* is a $\mathbb{T}$-automaton ($\bigstar$) for which

- $\mathbb{T}$ is the tape monad over Γ;
- 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 $p(i, \sigma_{+j}) = p(i + j, \sigma)$ if $\sigma =_{i \pm k} \sigma'$;
- $a^m : TB \to B$ is given by evaluation as in Definition 5.4.

It can be shown that tape $\mathbb{T}$-automata over A_τ are equivalent to 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 8.6. *For every tape automaton m over A_τ, Γ with $|\Gamma| \geq 2$ containing a special blank symbol $\square$, and every state $x \in X$ the following language is recursively enumerable: $\{w \in A^* \mid [\![x]\!]^\tau_m(w)(0, \sigma_\square) = 1\}$, where $\sigma_\square$ is the constant function on $\square$. Conversely, every recursively enumerable language can be represented in this way.*

9 Conclusions and Future Work

In this paper, we have presented the first steps towards a coalgebraic Chomsky hierarchy. We have given a coalgebraic account of machines, languages and expressions and presented several results of our theory including a generic Kleene-style theorem (Theorem 4.9) and one-direction of a Chomsky-Schützenberger-style theorem (Theorem 7.3). We have also given the first treatment of Turing machines in a coalgebraic setting: the

observational trace semantics of tape automata yields precisely the recursively enumerable languages.

There are several possible directions for future work. We plan to derive a sound calculus of reactive expressions extending [3] and explore the boundaries for completeness (by Corollary 6.6 completeness is only possible for specific choices of $\mathbb{T}$); establish a converse to Theorem 7.3;capture further language and complexity classes, such as the context-sensitive languages. Capturing various classes of machines under the umbrella of coalgebra results 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 would like to thank anonymous referees for their suggestions on improving the presentation and for pointing out relevant literature.

References

[1] Haskell 98 Language and Libraries — The Revised Report. Cambridge University Press (2003), also: J. Funct. Prog. **13** (2003)
[2] Adámek, J., Milius, S., Velebil, J.: Iterative algebras at work. Math. Structures Comput. Sci. 16(6), 1085–1131 (2006)
[3] Bonsangue, M.M., Milius, S., Silva, A.: Sound and complete axiomatizations of coalgebraic language equivalence. ACM Trans. Comput. Log. 14(1), 7:1–7:52 (2013)
[4] Book, R.V., Greibach, S.A.: Quasi-realtime languages. Mathematical Systems Theory 4(2), 97–111 (1970)
[5] Brzozowski, J.A.: Derivatives of regular expressions. J. ACM 11(4), 481–494 (1964)
[6] Coumans, D., Jacobs, B.: Scalars, monads, and categories. In: Sadrzadeh, C.H.M., Grefenstette, E. (eds.) Quantum physics and linguistics. A compositional, diagrammatic discourse. pp. 184–216. Oxford University Press (2013)
[7] Courcelle, B.: Fundamental properties of infinite trees. Theoretical Computer Science 25(2), 95 – 169 (1983)
[8] Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. Springer (2009)
[9] Eilenberg, S.: Automata, Languages, and Machines, Pure and Applied Mathematics, vol. A. Academic Press (1974)
[10] Fiore, M.P., Moggi, E., Sangiorgi, D.: A fully abstract model for the π-calculus. Inf. Comput. 179(1), 76–117 (2002)
[11] Freyd, P.: Algebra valued functors in general and tensor products in particular. Colloq. Math. 14, 89–106 (1966)
[12] Goguen, J.A., Thatcher, J.W., Wagner, E.G., Wright, J.B.: Initial algebra semantics and continuous algebras. J. ACM 24(1), 68–95 (Jan 1977)
[13] Goncharov, S.: Trace semantics via generic observations. In: Heckel, R., Milius, S. (eds.) CALCO 2013. LNCS, vol. 8089 (2013)
[14] Hyland, M., Levy, P.B., Plotkin, G.D., Power, J.: Combining algebraic effects with continuations. Theor. Comput. Sci. 375(1-3), 20–40 (2007)
[15] Jacobs, B.: A bialgebraic review of deterministic automata, regular expressions and languages. In: Futatsugi, K., Jouannaud, J.P., Meseguer, J. (eds.) Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday. LNCS, vol. 4060, pp. 375–404 (2006)
[16] Jacobs, B.: Coalgebraic trace semantics for combined possibilitistic and probabilistic systems. Electr. Notes Theor. Comput. Sci. 203(5), 131–152 (2008)

[17] Jacobs, B.: Convexity, duality and effects. In: Proc. TCS 2010. IFIP AICT, vol. 323, pp. 1–19 (2010)
[18] Jacobs, B., Silva, A., Sokolova, A.: Trace semantics via determinization. In: CMCS'12, LNCS, vol. 7399, pp. 109–129. Springer (2012)
[19] Kambites, M.: Formal languages and groups as memory. Communications in Algebra 37(1), 193–208 (2009)
[20] Kelly, G.M.: A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on. Bull. Austral. Math. Soc. 22, 1–83 (1980)
[21] Klin, B.: Bialgebras for structural operational semantics: An introduction. Theor. Comput. Sci. 412(38), 5043–5069 (2011)
[22] Kock, A.: On double dualization monads. Math. Scand. 27, 151–165 (1970)
[23] Lawson, M.V.: Inverse Semigroups: The Theory of Partial Symmetries. World Scientific Publishing Company (1999)
[24] Mac Lane, S.: Categories for the Working Mathematician. Springer (1971)
[25] Milius, S.: A sound and complete calculus for finite stream circuits. In: Proc. 25th Annual Symposium on Logic in Computer Science (LICS'10). pp. 449–458. IEEE Computer Society (2010)
[26] Moggi, E.: Notions of computation and monads. Inf. Comput. 93, 55–92 (1991)
[27] Myers, R.: Rational Coalgebraic Machines in Varieties: Languages, Completeness and Automatic Proofs. Ph.D. thesis, Imperial College London (2013)
[28] Plotkin, G., Power, J.: Notions of computation determine monads. In: FoSSaCS'02. LNCS, vol. 2303, pp. 342–356. Springer (2002)
[29] Power, J., Shkaravska, O.: From comodels to coalgebras: State and arrays. In: CMCS'04. ENTCS, vol. 106, pp. 297–314 (2004)
[30] Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. 3(2), 114–125 (Apr 1959)
[31] Rabin, M.O.: Probabilistic automata. Information and Control 6(3), 230–245 (1963)
[32] Render, E., Kambites, M.: Rational subsets of polycyclic monoids and valence automata. Information and Computation 207(11), 1329 – 1339 (2009)
[33] Rozenberg, G., Salomaa, A. (eds.): Handbook of formal languages, vol. 1: Word, Language, Grammar. Springer-Verlag New York, Inc. (1997)
[34] Rutten, J.: Universal coalgebra: A theory of systems. Theoret. Comput. Sci. 249, 3–80 (2000)
[35] Rutten, J.J.M.M.: Behavioural differential equations: A coinductive calculus of streams, automata, and power series. Theor. Comput. Sci. 308(1-3), 1–53 (2003)
[36] Segala, R.: Modelling and Verification of Randomized Distributed Real-Time Systems. Ph.D. thesis, Massachusetts Institute of Technology (1995)
[37] Segala, R., Lynch, N.A.: Probabilistic simulations for probabilistic processes. Nordic Journal of Computing 2(2), 250–273 (1995)
[38] Silva, A.: Kleene coalgebra. Ph.D. thesis, Radboud Univ. Nijmegen (2010)
[39] Silva, A., Bonchi, F., Bonsangue, M., Rutten, J.: Generalizing determinization from automata to coalgebras. LMCS 9(1) (2013)
[40] Syme, D., Granicz, A., Cisternino, A.: Expert F#. Apress (2007)
[41] Turi, D., Plotkin, G.D.: Towards a mathematical operational semantics. In: Proc. Logic in Computer Science (LICS) (1997)
[42] Varacca, D., Winskel, G.: Distributing probability over non-determinism. Math. Struct. Comput. Sci. 16, 87–113 (2006)
[43] Winter, J., Bonsangue, M.M., Rutten, J.J.M.M.: Coalgebraic characterizations of context-free languages. LMCS 9(3) (2013)

A Appendix

In this appendix we provide proofs for all the results of our paper.

A.1 Proof of Proposition 2.6

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 $u\sigma$ where u is a subexpression of e and σ is a substitution sending free variables of u to closed subexpressions of e. Then, S is closed under a-derivatives, for $\partial_{a_i}(u\sigma) = u_i\sigma[u/x]$ whenever $u = \mu x.(a_1.u_1 \Cap \cdots a_n.u_n \Cap c)$ and $\partial_a(u\sigma) = \partial_a(\sigma(x))$ if u is a variable x which lies in S by the previous case since $\sigma(x)$ is a closed subexpression of e and so must start with a μ-operator. By definition, $e \in S$, hence $E \subseteq S$. Since S is finite, so is E. □

A.2 Proof of Theorem 3.7

Observe that the following equation belongs to the tape theory

$$write_i(lmove^k(write_j(rmove^k(x)))) = lmove^k(write_j(rmove^k(write_i(x)))) \tag{A.1}$$

for any i, j, k.

Given a finite set of identities $\mathcal{A}$ belonging to the tape theory, we construct a model M of $\mathcal{A}$ which does not satisfy (A.1) and therefore prove the claim.

Let m be larger than the number of instances of operations $lmove$ and $rmove$ in any term from $\mathcal{A}$ occurring as the left- or the right-hand side of an identity. Our model M is carried by the finite ring $\mathbb{Z}_m$ of congruence classes modulo m. We interpret the operations of the tape theory over the set of endomaps $\mathbb{Z}_m \times \Gamma^{\mathbb{Z}_m} \to \mathbb{Z}_m \times \Gamma^{\mathbb{Z}_m}$ as follows:

$$\begin{aligned}
[\![read]\!](p_1, \ldots, p_n)(z, \sigma) &= p_{\sigma(z)}(z, \sigma) \\
[\![write_i]\!](p)(z, \sigma) &= p(z, \sigma[z \mapsto \gamma_i]) \\
[\![lmove]\!](p)(z, \sigma) &= p(z - 1, \sigma) \\
[\![rmove]\!](p)(z, \sigma) &= p(z + 1, \sigma)
\end{aligned}$$

where i ranges from 1 to n and $+$ and $-$ are arithmetic operations modulo m, i.e. the operations of $\mathbb{Z}_m$.

Let $p = q$ be an equation from $\mathcal{A}$. Then for any $z \in \mathbb{Z}$ and any $\sigma \in \Gamma^{\mathbb{Z}}$, $[\![p]\!]_{T1}(z, \sigma) = [\![q]\!]_{T1}(z, \sigma)$. By assumption, both p and q can update at most m cells in a row and the cells that are modified by p must agree with those ones modified by q. Therefore, for any z there exist $a_z, b_z \in \mathbb{Z}$ such that $|a_z - b_z| \leq m$, $a_z \leq z \leq b_z$ and $\mathsf{pr}_2([\![p]\!]_{T1}(z, \sigma))(i) = \mathsf{pr}_2([\![q]\!]_{T1}(z, \sigma))(i) = \sigma(i)$ for any $\sigma \in \Gamma^{\mathbb{Z}}$ and any i outside the interval $[a, b]$. Given $\rho \in \Gamma^{\mathbb{Z}_m}$ and $0 \leq z \leq m - 1$, let $\rho_* \in \Gamma^{\mathbb{Z}}$ be defined as follows:

$$\begin{aligned}
&\rho_*(i) = \rho(i \bmod m) && (a_z \leq i \leq b_z) \\
&\rho_*(i) = \gamma_n && (\text{otherwise})
\end{aligned}$$

Let $[\![p]\!]_{T1}(z,\rho_*) = [\![q]\!]_{T1}(z,\rho_*) = \langle z', \rho'_*\rangle$. It is then easy to see that in M we have $[\![p]\!](z,\rho) = [\![q]\!](z,\rho) = \langle z' \bmod m, \rho'\rangle$ where $\rho'(i) = \rho'_*(i)$, $i = 0,\ldots,m$. We have thus shown that $\mathcal{A}$ is valid over M. Now, if we take $k = m$ in (A.1) we obtain for $i \neq j$ that

$$\begin{aligned} & [\![write_i(lmove^m(write_j(rmove^m(x))))]\!](0,\sigma) \\ = \ & \sigma[0 \mapsto j] \\ \neq \ & \sigma[0 \mapsto i] \\ = \ & [\![lmove^m(write_j(rmove^m(write_i(x))))]\!](0,\sigma) \end{aligned}$$

and thus the proof is finished. □

A.3 Proof of Theorem 4.8

Recall first that $\mathbb{T}$-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-coalgebra structure is a Σ-algebra morphism since it is a $\mathbb{T}$-algebra morphism). Thus, the trace 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:

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

Since (the lifting of) L preserves monos we obtain an L-coalgebra 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

$$[\![f]\!]_\Sigma\big([t_1]_\sim,\ldots,[t_k]_\sim\big) = \big[[\![f]\!]_\Sigma(t_1,\ldots,t_k)\big]_\sim.$$

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

A.4 Proof of Theorem 4.9

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

We assume w.l.o.g. that μ-operators in e do not bind the same variable twice; this can be ensured by α-conversion. Let $\{x_1, \ldots, x_n\}$ be the set of all variables occurring in e. For every $i = 1, \ldots, n$, let

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

be a subterm of e where the t_j^i are Σ-terms and each θ_j^i substitutes every variable x_k free in t_j^i with t_k. Note that the t_i, the θ_j^i and the t_j^i are uniquely determined by e.

Starting with the pair

$$([\,], \{t_i \mid t_i \text{ is a maximal } \gamma\text{-subexpression in } e\})$$

where $[\,]$ denotes the empty substitution we successively define pairs of the form (σ, S) where S is a set of γ-subexpressions of e and σ is a substitution replacing some variables x_i with reactive expressions. Unless S is empty, the successor (σ', S') of (σ, S) is (nondeterministilly) defined as follows. Let $t_i \in S$. Then σ' is the completion of σ by the clause $x_i \mapsto t_i\sigma$ and S' is obtained by the removal of t_i and addition of the maximal γ-subterms of t_i. This procedure eventually terminates, for each step removes one application of μ. In the end we obtain a map sending each x_i to a closed γ-expression, which we denote e_i, i.e., $\sigma = [e_1/x_1, \ldots, e_n/x_n]$.

We assume henceforth the representation

$$e_i = \mu x_i.\, (a_1.e_1^i \pitchfork \cdots \pitchfork a_n.e_n^i \pitchfork b_i).$$

Observe that, by construction, $e_i = t_i\sigma$. Moreover, we have

$$\begin{aligned} e_i &= \mu x_i.\, (a_1.e_1^i \pitchfork \cdots \pitchfork a_n.e_n^i \pitchfork b_i) \\ &= t_i\sigma = \mu x_i.\, (a_1.t_1^i\theta_1^i\sigma_{-i} \pitchfork \cdots \pitchfork a_n.t_n^i\theta_n^i\sigma_{-i} \pitchfork b_i) \end{aligned}$$

where σ_{-i} agrees with σ except that it leaves x_i unchanged. For any i, j we obtain that $t_j^i\theta_j^i\sigma_{-i} = e_j^i$ and therefore $e_j^i[e_i/x_i] = t_j^i\theta_j^i\sigma$. Recall that θ_j^i sends x_k to t_k and $t_k\sigma = e_k$, which results in $t_j^i\theta_j^i\sigma = t_j^i\sigma$. We have thus obtained

$$e_j^i[e_i/x_i] = t_j^i[e_1/x_1, \ldots, e_n/x_n].$$

This allows us to write the following definitions for o and ∂:

$$\begin{aligned} o(t(e_1, \ldots, e_n)) &= t^B(b_1, \ldots, b_n), \\ \partial_{a_j}(t(e_1, \ldots, e_n)) &= t(t_j^1[e_1/x_1, \ldots, e_n/x_n], \ldots, t_j^n[e_1/x_1, \ldots, e_n/x_n]), \end{aligned}$$

for any Σ-term t. Let $\sigma_{a_j} = [t_j^1/x_1, \ldots, t_j^n/x_n]$ and let by induction $\sigma_\epsilon = \mathrm{id}$, $\sigma_{a_j w} = \sigma_{a_j}\sigma_w$. By further induction we obtain

$$o(\partial_w(t(e_1, \ldots, e_n))) = (t\sigma_w)^B(b_1, \ldots, b_n). \tag{A.2}$$

Suppose, $e = s(e_{n_1}, \dots, e_{n_k})$ and let $X = \{s(x_{n_1}, \dots, x_{n_k}), x_1, \dots, x_n\}$. We will now turn TX into a reactive $\mathbb{T}$-algebra. Notice that every element of TX can be written as $[t(x_1, \dots, x_n)]_\equiv$, where t is a Σ-term in the variables x_i and $[t]_\equiv$ denotes the equivalence class in TX. Now define

$$o([t(x_1, \dots, x_n)]_\equiv) = t^B(b_1, \dots, b_n)),$$
$$\partial_{a_j}([t(x_1, \dots, x_n)]_\equiv) = \left[t(t_j^1, \dots, t_j^n)\right]_\equiv.$$

It is not difficult to see that the $\mathbb{T}$-algebra and the L-coalgebra structures interact properly, i.e. o and the ∂_a are $\mathbb{T}$-algebra morphisms; in fact, $o = \alpha \cdot Tf$, where $\alpha : TB \to B$ is the $\mathbb{T}$-algebra on B and the map $f : X \to B$ is defined by $f(x_i) = b_i$, $i = 1, \dots, n$; and $\partial_{a_j} = g_j^\dagger$ where $g_j : X \to TX$ is the map defined by $g_j(x_i) = [t_j^i]_\equiv$.

We have $\partial_{a_j}([t]_\equiv) = [t\sigma_{a_j}]_\equiv$, so an easy induction shows that

$$o(\partial_w([t(x_1, \dots, x_n)]_\equiv)) = (t\sigma_w)^B(b_1, \dots, b_n).$$

By comparing this to (A.2) we obtain by Proposition 2.2, $[\![t(e_1, \dots, e_n)]\!] = [\![t(x_1, \dots, x_n)]\!]$ and in particular $[\![e]\!] = [\![s(x_{n_1}, \dots, x_{n_k})]\!]$. By Remark 4.5, the constructed reactive $\mathbb{T}$-algebra is equivalent to a $\mathbb{T}$-automaton.

Let us show the converse implication. Suppose, we are given a $\mathbb{T}$-automaton (★). The generalized powerset construction introduces a reactive $\mathbb{T}$-algebra over TX for which

$$\begin{aligned} o([t(x_1, \dots, x_n)]_\equiv) &= t^B(b_1, \dots, b_n) \\ \partial_{a_j}([t(x_1, \dots, x_n)]_\equiv) &= \left[t(t_j^1, \dots, t_j^n)\right]_\equiv \end{aligned} \tag{A.3}$$

where $b_i = o^m(x_i) \in B$, t_j^i is a term representing $t^m(a_j, x_i) \in TX$ and $[t]_\equiv$ denotes the equivalence class of the Σ-term t in TX. Let c_i be a Σ-term over B_0 such that $b_i = \mathsf{ev}(c_i)$, where ev is the map evaluating a Σ-term over B_0 in B. We successively build expressions $u_n, \dots, u_1$ 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

$$u_n = \mu x_n.\, (a_1.t_1^n \pitchfork \cdots \pitchfork a_n.t_n^n \pitchfork c_n)$$
$$u_i = \mu x_i.\, (a_1.t_1^i[u_{i+1}/x_{i+1}, \dots, u_n/x_n] \pitchfork \cdots \pitchfork a_n.t_n^i[u_{i+1}/x_{i+1}, \dots, u_n/x_n] \pitchfork c_i)$$

for all $i = n-1, \dots, 1$, 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. Notice that the Σ-terms t_j^i in the construction are precisely the t_j^i from (A.3) that we used to define the expressions u_i. Now the construction yields further expressions e_i, $i = 2, \dots, n$ and for every i expressions $e_1^i, \dots, e_n^i$ satisfying the identities

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

By the same argument as in the first part of the proof we obtain (A.2). On the other hand, for the original reactive $\mathbb{T}$-algebra, also

$$o(\partial_w([t(x_1, \dots, x_n)]_\equiv)) = (t\sigma_w)^B(b_1, \dots, b_n)$$

and therefore we are done by Proposition 2.2. □

A.5 Additive reactive expressions

In this section, we prove the following claim, which appears just before Example 5.2.

> *Given a reactive expression we obtain an additive reactive expression by replacing recursively each $\pitchfork$ with $+$. Conversely, any additive reactive expression can be transformed to a reactive expression. The latter transformation is inverse to the former modulo $\sim$.*

Let RExp and AExp denote the sets of reactive and additive reactive expressions, respectively.

First, we observe that AExp carries a Σ-algebra structure. Moreover it also carries an L-coalgebra structure given by $o\colon \mathsf{AExp} \to B$ and, for all $a \in A$, $\partial_a\colon \mathsf{AExp} \to \mathsf{AExp}$:

$$\begin{aligned} o(b) &= b & o(\mu x.\,\gamma) &= o(\gamma[\mu x.\gamma/x]) \\ o(a.\gamma) &= \emptyset & o(f(\gamma_1,\dots,\gamma_n)) &= f^B(o(\gamma_1),\dots,o(\gamma_n)) \end{aligned}$$

$$\begin{aligned} \partial_a(b) &= \emptyset \\ \partial_a(\mu x.\gamma) &= \partial_a(\gamma[\mu x.\gamma/x]) \\ \partial_a(a'.\gamma) &= \begin{cases} \gamma & \text{if } a = a' \\ \emptyset & \text{otherwise} \end{cases} \\ \partial_a(f(\gamma_1,\dots,\gamma_n)) &= f(\partial_a(\gamma_1),\dots,\partial_a(\gamma_n)) \end{aligned}$$

Clearly, o and ∂_a are also Σ-algebra morphisms. The coalgebra structure on AExp allows us to define semantics of these expressions by finality via $[\![-]\!]\colon \mathsf{AExp} \to B^{A^*}$.

We now define the translation maps between RExp and AExp and show that these translations preserve the semantics.

Let $\overline{\mathsf{tr}}\colon \mathsf{RExp} \to \mathsf{AExp}$ be the map defined by

$$\begin{aligned} \overline{\mathsf{tr}}(f(\gamma_1,\dots,\gamma_n)) &= f(\overline{\mathsf{tr}}(\gamma_1),\dots,\overline{\mathsf{tr}}(\gamma_n)) \\ \overline{\mathsf{tr}}(\mu x.\ (a_1.\delta_1 \pitchfork \cdots \pitchfork a_n.\delta_n \pitchfork b)), &= \mu x.\ (a_1.\overline{\mathsf{tr}}(\delta_1) + \cdots + a_n.\overline{\mathsf{tr}}(\delta_n) + b)\,. \end{aligned}$$

and $\mathsf{tr}\colon \mathsf{AExp} \to \mathsf{RExp}$ be the map defined by

$$\begin{aligned} \mathsf{tr}(b) &= \mu x.\,(a_1.\emptyset \pitchfork \cdots \pitchfork a_n.\emptyset \pitchfork b), \\ \mathsf{tr}(a_i.\gamma) &= \mu x.\,(a_1.\emptyset \pitchfork \cdots \pitchfork a_i.\mathsf{tr}(\gamma) \pitchfork \cdots \pitchfork a_n.\emptyset \pitchfork \emptyset), \\ \mathsf{tr}(f(\gamma_1,\dots,\gamma_n)) &= f(\mathsf{tr}(\gamma_1),\dots,\mathsf{tr}(\gamma_n)), \\ \mathsf{tr}(\mu x.\gamma) &= \mu x.\,a_1.\mathsf{tr}(\partial_{a_1}(\mu x.\gamma)) \pitchfork \cdots \pitchfork a_n.\mathsf{tr}(\partial_{a_n}(\mu x.\gamma)) \pitchfork o(\mu x.\gamma). \end{aligned}$$

We now show that $[\![e]\!] = [\![\mathsf{tr}(e)]\!]$, for $e \in \mathsf{AExp}$ and $[\![e']\!] = [\![\overline{\mathsf{tr}}(e')]\!]$, for all $e' \in \mathsf{RExp}$.

It is sufficient to show that:

$$o(e) = o(\mathsf{tr}(e)) \qquad o(e') = o(\overline{\mathsf{tr}}(e')) \tag{A.4}$$

and

$$\mathsf{tr}(\partial_a(e)) = \partial_a(\mathsf{tr}(e)) \qquad \overline{\mathsf{tr}}(\partial_a(e')) = \partial_a(\overline{\mathsf{tr}}(e')) \tag{A.5}$$

These equations state that tr and $\overline{\mathsf{tr}}$ are L-coalgebra homomorphisms. Therefore, by finality of B^{A^*} we have

$$[\![-]\!] = (\,\mathsf{AExp} \xrightarrow{\mathsf{tr}} \mathsf{RExp} \xrightarrow{[\![-]\!]} B^{A^*}\,) \quad \text{and} \quad [\![-]\!] = (\,\mathsf{RExp} \xrightarrow{\overline{\mathsf{tr}}} \mathsf{AExp} \xrightarrow{[\![-]\!]} B^{A^*}\,),$$

which completes the proof. Let us prove (A.4) and (A.5). This is done by structural induction on e and e', respectively.

$$\begin{aligned}
o(\mathsf{tr}(b)) &= o(\mu x.\, a_1.\emptyset \pitchfork \cdots \pitchfork a_n.\emptyset \pitchfork b) = b = o(b)\\
o(\mathsf{tr}(a_i.\gamma)) &= o(\mu x.\, a_1.\emptyset \pitchfork \cdots \pitchfork a_i.\mathsf{tr}(\gamma) \pitchfork \cdots \pitchfork a_n.\emptyset \pitchfork \emptyset)\\
&= \emptyset = o(a_i.\gamma)\\
o(\mathsf{tr}(f(\gamma_1,\ldots,\gamma_n))) &= o(f(\mathsf{tr}(\gamma_1),\ldots,\mathsf{tr}(\gamma_n)))\\
&= f^B(o(\mathsf{tr}(\gamma_1)),\ldots,o(\mathsf{tr}(\gamma_1)))\\
&= f^B(o(\gamma_1),\ldots,o(\gamma_n))\\
&= o(f(\gamma_1,\ldots,\gamma_n))\\
o(\mathsf{tr}(\mu x.\gamma)) &= o(\mu x.\, a_1.\mathsf{tr}(\partial_{a_1}(\mu x.\gamma)) \pitchfork \cdots \pitchfork o(\mu x.\gamma))\\
&= o(\mu x.\gamma)
\end{aligned}$$

$$\begin{aligned}
o(\overline{\mathsf{tr}}(f(\gamma_1,\ldots,\gamma_n))) &= o(f(\overline{\mathsf{tr}}(\gamma_1),\ldots,\overline{\mathsf{tr}}(\gamma_n)))\\
&= f^B(o(\overline{\mathsf{tr}}(\gamma_1)),\ldots,o(\overline{\mathsf{tr}}(\gamma_1)))\\
&= f^B(o(\gamma_1),\ldots,o(\gamma_n))\\
&= o(f(\gamma_1,\ldots,\gamma_n))\\
o(\overline{\mathsf{tr}}(\mu x.\ (a_1.\delta_1 \pitchfork \cdots \pitchfork a_n.\delta_n \pitchfork b))) &= o(\mu x.\ (a_1.\overline{\mathsf{tr}}(\delta_1) + \cdots + a_n.\overline{\mathsf{tr}}(\delta_n) + b))\\
&= b = o(\mu x.\ (a_1.\delta_1 \pitchfork \cdots \pitchfork a_n.\delta_n \pitchfork b))
\end{aligned}$$

$$\begin{aligned}
\partial_a(\mathsf{tr}(b)) &= \emptyset = \mathsf{tr}(\emptyset) = \mathsf{tr}(\partial_a(b))\\
\partial_a(\mathsf{tr}(a_i.\gamma)) &= \partial_a(\mu x.\, a_1.\emptyset \pitchfork \cdots a_i.\mathsf{tr}(\gamma) \cdots \pitchfork a_n.\emptyset \pitchfork \emptyset)\\
&= \left\{\begin{array}{ll} \mathsf{tr}(\gamma) & \text{if } a = a_i \\ \emptyset & \text{otherwise} \end{array}\right\} = \mathsf{tr}(\partial_a(a_i.\gamma))\\
\partial_a(\mathsf{tr}(f(\gamma_1,\ldots,\gamma_n))) &= \partial_a(f(\mathsf{tr}(\gamma_1),\ldots,\mathsf{tr}(\gamma_n)))\\
&= f(\partial_a(\mathsf{tr}(\gamma_1)),\ldots,\partial_a(\mathsf{tr}(\gamma_1)))\\
&= f(\gamma_1,\ldots,\gamma_n)\\
\partial_a(\mathsf{tr}(\mu x.\gamma)) &= \partial_a(\mu x.\, a_1.\mathsf{tr}(\partial_{a_1}(\mu x.\gamma)) \pitchfork \cdots \pitchfork o(\mu x.\gamma))\\
&= \mathsf{tr}(\partial_a(\mu x.\gamma))
\end{aligned}$$

$$\begin{aligned}
\partial_a(\overline{\mathsf{tr}}(\mu x.\ (a_1.\delta_1 \pitchfork \cdots \pitchfork a_n.\delta_n \pitchfork b))) &= \partial_a(\mu x.\ (a_1.\overline{\mathsf{tr}}(\delta_1) + \cdots + a_n.\overline{\mathsf{tr}}(\delta_n) + b))\\
&= \left\{\begin{array}{ll} \overline{\mathsf{tr}}(\delta_i) & \text{if } a = a_i \\ \emptyset & \text{otherwise} \end{array}\right\}\\
&= \overline{\mathsf{tr}}(\partial_a(\mu x.\ (a_1.\delta_1 \pitchfork \cdots \pitchfork a_n.\delta_n \pitchfork b)))
\end{aligned}$$

□

A.6 Proof of Theorem 5.5

We begin the proof with a technical lemma. First we recall how the trace semantics for $\mathbb{T}$-automata are defined. Let $\iota : B^{A^*} \to B \times (B^{A^*})^A$ be the final L-coalgebra. We obtain the following diagram:

$$\begin{array}{ccccc} X & \xrightarrow{\eta} & TX & \xrightarrow{\widehat{m}^\sharp} & B^{A^*} \\ {\scriptstyle m}\downarrow & \swarrow_{m^\sharp} & & & \downarrow{\scriptstyle \iota} \\ B \times (TX)^A & & \xrightarrow{\mathrm{id} \times (\widehat{m}^\sharp)^A} & & B \times (B^{A^*})^A \end{array} \tag{A.6}$$

where $\widehat{m}^\sharp$ is the final coalgebra morphism. We obtain thus the semantics map

$$[\![-]\!]_m = \left(X \xrightarrow{\eta} TX \xrightarrow{\widehat{m}^\sharp} B^{A^*} \right).$$

We shall need an explicit description of the action of this semantics map in terms of the given data of a $\mathbb{T}$-automaton.

Lemma A.1. *Given any $\mathbb{T}$-automaton (★), $x \in X$ and $w \in A^*$ then*

$$\begin{aligned} [\![x]\!]_m(\epsilon) &= o^m(x) \\ [\![x]\!]_m(a \cdot u) &= a^m \left(\mathsf{do}\ y \leftarrow t^m(a,x); \eta [\![y]\!]_m(u) \right) \end{aligned}$$

Proof. Recall that the final coalgebra structure ι arises from $o : B^{A^*} \to B$ and $\partial_a : B^{A^*} \to B^{A^*}$ with $o(\sigma) = \sigma(\epsilon)$ and $\partial_a(\sigma) = \lambda w.\sigma(aw)$. The commutativity of (A.6) can now equivalently be restated as the two equations

$$\begin{aligned} o([\![x]\!]_m) &= o^m(x), \\ \partial_a([\![x]\!]_m) &= \widehat{m}^\sharp(t^m(x,a)). \end{aligned}$$

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

$$\widehat{m}^\sharp = \alpha \cdot T[\![-]\!]_m,$$

where α is the $\mathbb{T}$-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

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

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

$$\widehat{m}^\sharp(-)(u) = (TX \xrightarrow{T([\![-]\!]_m(u))} TB \xrightarrow{a^m} B);$$

indeed we have:

$$\begin{aligned}\widehat{m}^\sharp(-)(u) &= \mathsf{ev}_u \cdot \widehat{m}^\sharp \\ &= \mathsf{ev}_u \cdot \alpha \cdot T[\![-]\!]_m \\ &= a^m \cdot T\mathsf{ev}_u \cdot T[\![-]\!]_m \\ &= a^m \cdot T([\![-]\!]_m(u)).\end{aligned}$$

Now we can compute:

$$\begin{aligned}[\![x]\!]_m(au) &= \partial_a([\![x]\!]_m)(u) && \text{definition of } \partial_a \\ &= \widehat{m}^\sharp(t^m(x,a))(u) && \text{by (A.6)} \\ &= (a^m \cdot T([\![-]\!]_m)(u))(t^m(x,a)).\end{aligned}$$

And the last line is exactly the desired right-hand side of the second statement. □

Now we proceed to the proof of Theorem 5.5. We recall that a deterministic pushdown automaton (dpda) M is determined by a transition function

$$\delta : Q \times (A + \{\epsilon\}) \times \Delta \to Q \times \Delta^* + \{\bot\}, \tag{A.7}$$

an initial stack symbol $\square \in \Delta$, 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 any $x \in Q, \gamma \in \Delta$ (exclusively) either $\delta(x,\epsilon,\gamma) \neq \bot$ or $\delta(x,a,\gamma) \neq \bot$ for all $a \in A$; whenever $\delta(x,a,\gamma) \neq \bot$ with $a \in A + \{\epsilon\}$, the second component of $\delta(x,a,\square)$ has the form $s'\square$. 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 consumes w, starts at $\langle x_0, \square\rangle$ 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 [**?**].

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 such $n \geq 1$ 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. (The numbers $n_{x,a}$ and n_x exist by the definition of the stack monad.) As the state space of M we take

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

where $m = \max\{n_x, \max_a n_{a,x}\}$. Let $\Delta = \Gamma + \{\square\}$. 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 $\gamma \neq \square$ and $|s| < m$;
(ii) $\delta(\langle x, s\rangle, \epsilon, \square) = \langle\langle x, s\square\rangle, \square\rangle$ if $|s| < m$;
(iii) $\delta(\langle x, s\square^k\rangle, a, \gamma) = \langle\langle y, \epsilon\rangle, s'\gamma\rangle$ for any $a \in A$ if $\gamma \in \Delta$ where $s \in \Gamma^{m-k}$ and $\langle y, s'\rangle = t^{\mathcal{m}}(x, a)(s)$.

Finally, let

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

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 $\mathcal{m}$, 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 that by allowing the completion of the second component of the state with an appropriate number of symbols $\square$ added from the right if the stack happens to be shorter than m.

Our goal is to show that for any $w \in A^*$, $[\![x_0]\!]_{\mathcal{m}}(w)(\gamma_0) = 1$ 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 $s \in \Gamma^*$, $[\![x]\!]_{\mathcal{m}}(w)(s) = 1$ iff there is a chain of transitions C over configurations of M corresponding to w, starting at $\langle\langle x, \epsilon\rangle, s\square\rangle$ and finishing in an accepting state. We proceed by induction over the length of w.

- Let $w = \epsilon$. Then by Lemma A.1 $[\![x]\!]_{\mathcal{m}}(w)(s) = o^{\mathcal{m}}(x)(s) = o^{\mathcal{m}}(x)(s')$ where s' is obtained from s by truncating the latter on the right down to the size not exceeding m. Therefore, $[\![x]\!]_{\mathcal{m}}(w)(s) = 1$ iff $\langle x, s'\square^k\rangle \in Q$ belongs to F with $k = m - |s'|$. On the other hand, by (i)–(ii) any chain C of transitions corresponding to $w = \epsilon$ and starting at $\langle\langle x, \epsilon\rangle, s\square\rangle$ must be a prefix of the following chain:

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

  where $s = s'u$ and $k = m - |s'|$. Clearly, C leads to an accepting configuration iff $\langle x, s'\square^k\rangle$ is an accepting state.
- Let $w = au$. Then by Lemma A.1,

  $$\begin{aligned}[\![x]\!]_{\mathcal{m}}(w)(s) &= a^{\mathcal{m}}\left(\mathsf{do}\ y \leftarrow t^{\mathcal{m}}(x, a); \eta[\![y]\!]_{\mathcal{m}}(u)\right)(s)\\ &= \mathsf{let}\ \langle y, s'\rangle = t^{\mathcal{m}}(x, a)(s)\ \mathsf{in}\ [\![y]\!]_{\mathcal{m}}(u)(s').\end{aligned}$$

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

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

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

C' by prepending C with

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

Conversely, given a chain of transitions C' for w from $\langle\langle x, \epsilon\rangle, s\Box\rangle$ and leading to a final state, then it must be a chain C starting at $\langle\langle y, \epsilon\rangle, s'\Box\rangle$ 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 (A.7), an initial state $q_0 \in Q$, a set of accepting states $F \subseteq Q$ and an initial stack symbol $\Box$. Let us define an $\mathbb{T}$-automaton (★) with $X = Q + \{\bot\}$ and $\mathbb{T}$ being a stack monad over Δ as follows: for any $q \in X$, $s \in \Delta^*$, $a \in A$, $o^m(q)(s) = 1$ iff $q \in F$ and

$$\begin{aligned} t^m(q,a)(\epsilon) &= t^m(\bot, a)(\gamma s) = \langle \bot, \epsilon\rangle \\ t^m(q,a)(\gamma s) &= \big(\mathsf{let}\ \langle q', s'\rangle = \delta(q,a,\gamma)\ \mathsf{in}\ \langle q', s's\rangle\big) \qquad (q \neq \bot) \end{aligned}$$

Let us show by induction over the length of $w \in A^*$ that for any $q \in Q$, $s \in \Delta^*$ an acceptable 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 acceptable iff $q \in F$ iff $o^m(q)(s) = 1$. By Lemma A.1, the latter is equivalent to $[\![q]\!]_m(w)(s) = 1$.
- Let $w = au$. Then an acceptable 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 acceptable 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 A.1,

$$\begin{aligned} [\![q]\!]_m(w)(s) &= a^m\left(\mathsf{do}\ q' \leftarrow t^m(q,a); \eta\,[\![q']\!]_m(u)\right)(s) \\ &= \big(\mathsf{let}\ \langle q', s'\rangle = t^m(q,a)(s)\ \mathsf{in}\ [\![q']\!]_m(u)(s')\big), \end{aligned}$$

  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 (5.2) where $x_0 = q_0$ and $\gamma_0 = \Box$. □

A.7 Proof of Proposition 6.2

Let us refer to the stack theory over $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ as $\mathcal{E}$ and to the theory corresponding to the monad $\mathbb{T}$ as $\mathcal{T}$. We need to show that the category of $(\mathcal{E} \otimes \mathcal{T})$-algebras in $\mathbf{Set}$ is isomorphic to the Eilenberg-Moore category $\mathbf{Set}^{\mathbb{R}}$.

First we show that every $T(X \times \Gamma^*)^{\Gamma^*}$ can be turned into a $(\mathcal{E} \otimes \mathcal{T})$-algebra by putting

$$\begin{aligned} [\![push_i]\!]_{RX}(p)(s) &= p(\gamma_i s), \\ [\![pop]\!]_{RX}\langle p_1, \ldots, p_n, q\rangle(\gamma_i s) &= p_i(s), \end{aligned}$$

$$[\![pop]\!]_{RX}\langle p_1,\dots,p_n,q\rangle(\epsilon) = q(s),$$
$$[\![f]\!]_{RX}\langle p_1,\dots,p_k\rangle(s) = f^{T(X\times\Gamma^*)^{\Gamma^*}}(p_1(s),\dots,p_k(s)),$$

where i ranges from 1 to n and f ranges over the operations of $\mathcal{T}$. The axioms of $\mathcal{E}\otimes\mathcal{T}$ are verified routinely. E.g., for the first law of the stack theory we have

$$[\![push_i\, pop]\!]_{RX}\langle p_1,\dots,p_n,q\rangle(s) = [\![pop]\!]_{RX}\langle p_1,\dots,p_n,q\rangle(\gamma_i s) = p_i(s),$$

etc. The two tensor laws (for $push$ and for pop) are verified analogously.

It is then easy to see that RX, which is by definition a subset of $T(X\times\Gamma^*)^{\Gamma^*}$, is stable under the the operations of $\mathcal{E}\otimes\mathcal{T}$ and therefore itself forms a $(\mathcal{E}\otimes\mathcal{T})$-algebra. For example, let us verify the stability of RX under $push_i$. Suppose, $p\in RX$ is such that for some m

$$p(su) = \mathsf{do}\,\langle x,s'\rangle \leftarrow p(s); \eta\langle x, s'u\rangle \tag{A.8}$$

for all $s\in\Gamma^*$ with $|s|\geq m$. Then

$$\begin{aligned}&[\![push_i]\!]_{RX}(p)(su)\\ &\quad= p(\gamma_i su)\\ &\quad= \mathsf{do}\,\langle x,s'\rangle \leftarrow p(\gamma_i s); \eta\langle x, s'u\rangle\\ &\quad= \mathsf{do}\,\langle x,s'\rangle \leftarrow [\![push_i]\!]_{RX}(p)(s); \eta\langle x, s'u\rangle\end{aligned}$$

and hence $[\![push_i]\!]_{RX}(p)\in RX$.

We only need to prove that RX is the free $(\mathcal{E}\otimes\mathcal{T})$-algebra on X. First, we show that it is reachable from X, i.e. that every element $p\in RX$ is represented by a term over X. Let m be such that (A.8) holds whenever $|s|\geq m$ and proceed by induction over such m.

Suppose, $m=0$. Then by taking $s=\epsilon$ in (A.8), we obtain for any $u\in\Gamma^*$, $p(u) = \mathsf{do}\,\langle x,s'\rangle \leftarrow p(\epsilon); \eta\langle x, s'u\rangle$. Note that $p(\epsilon)\in T(X\times\Gamma^*)$ can be represented by a term over $X\times\Gamma^*$ in the theory $\mathcal{T}$. Then it follows that p is representable by the term

$$p(\epsilon)[\langle x,s\rangle \mapsto push_s(x)]$$

where for any $\alpha_1,\dots,\alpha_k\in\Gamma$, $push_{\alpha_1\cdots\alpha_k}(x)$ encodes iterated application of $push$ to x: $push_k(\cdots push_1(x)\cdots)$.

Suppose $m>0$ and let for any $\gamma\in\Gamma$, $p_\gamma(s) = p(\gamma s)$. Then, by (A.8) we have

$$p_\gamma(su) = \mathsf{do}\,\langle x,s'\rangle \leftarrow p_\gamma(s); \eta\langle x, s'u\rangle$$

whenever $|\gamma s|\geq m$, i.e. $|s|\geq m-1$. By induction, every p_γ is representable by a term, say t_γ. It is now easy to see that p is representable by the term

$$pop\big(t_{\gamma_1},\dots,t_{\gamma_n},p(\epsilon)[\langle x,s\rangle\mapsto push_s(x)]\big).$$

Finally, let us show that RX is free. Let t and r be two terms such that $[\![t]\!]_{RX} = [\![r]\!]_{RX}$ and let us show $\mathcal{E}\otimes\mathcal{T}\vdash t=r$. First we normalize t and r under the following rules:

$$push_i(pop(x_1,\dots,x_n,y))\to x_i$$

$$push_i(f(x_1,\ldots,x_k)) \to f(push_i(x_1),\ldots,push_i(x_k))$$

which are sound w.r.t. $\mathcal{E}\otimes\mathcal{T}$. Observe that in the normalized terms $push_i$ only occurs applied to variables. We now proceed by induction over the number m of pop operators in t and r.

Let $m=0$. Then

$$t = u(push_{s_1}(x_1),\ldots,push_{s_k}(x_k)) \quad \text{and} \quad r = h(push_{t_1}(y_1),\ldots,push_{t_l}(y_l))$$

where u and h are terms in the signature of $\mathcal{E}$ and $s_i, t_j \in \Gamma^*$. Let t' and r' be obtained from t and r by replacing any $push_s(z)$ with $\langle z, s\rangle$. Since $[\![t']\!]_{T(X\times\Gamma^*)} = [\![t]\!]_{RX}(\epsilon) = [\![r]\!]_{RX}(\epsilon) = [\![r']\!]_{T(X\times\Gamma^*)}$, we have $\mathcal{T}\vdash t'=r'$ and therefore $\mathcal{E}\otimes\mathcal{T}\vdash t'=r'$ which implies $\mathcal{E}\otimes\mathcal{T}\vdash t=r$.

Let $m>0$. For any i let us form t_i by the application of $push_i$ to t and further normalization of the result by the above rewrite rules. It is easy to see that the number of pop operators in any t_i is at most $m-1$. We also obtain a term t_ϵ as follows: we first rewrite t to the term t' according to the above rules and the following one:

$$\begin{array}{c} f(x_1,\ldots,x_{i-1},pop(y_1,\ldots,y_n,y),x_{i+1},\ldots,x_k) \\ \downarrow \\ pop\big(f(push_1(x_1),\ldots,push_1(x_{i-1}),y_1, \\ \quad push_1(x_{i+1}),\ldots,push_1(x_k)), \\ \vdots \\ f(push_n(x_1),\ldots,push_n(x_{i-1}),y_n, \\ \quad push_n(x_{i+1}),\ldots,push_n(x_k)), \\ f(x_1,\ldots,x_{i-1},y,x_{i+1},\ldots,x_k)\big). \end{array}$$

That this rule is sound is proved as follows: first replace in the "left-hand" term

$$f(\cdots, pop(\cdots),\cdots)$$

each variable x_j, $j=1,\ldots,i-1,i+1,\ldots,k$, using the first axiom of the stack theory by

$$pop(push_1(x_j),\ldots,push(x_j),x_j);$$

then apply the tensor law (see Definition 6.1) to the resulting term

$$f(pop(\cdots),\ldots,pop(\cdots))$$

to obtain the "right-hand" side of the rule.

Notice that t' contains exactly the same number of pop as t. If t contains no pop then let $t_\epsilon = t'$; else $t' = pop(t'_1,\ldots,t'_n,t_\epsilon)$ (which we use as implicit definition of t_ϵ). Now we obtain

$$\mathcal{E}\otimes\mathcal{T}\vdash t = pop(t_1,\ldots,t_n,t_\epsilon);$$

indeed, if t contains pop we reason as follows:

$$t = pop(push_1(t),\ldots,push_n(t),t)$$

$$
\begin{aligned}
&\quad \text{(by the 2nd axiom of the stack theory)}\\
&= pop(t_1, \ldots, t_n, t')\\
&\quad \text{(by rewriting)}\\
&= pop(t_1, \ldots, t_n, pop(t'_1, \ldots, t'_n, t_\epsilon))\\
&\quad \text{(using that } t' \text{ starts with } pop)\\
&= pop(t_1, \ldots, t_n, t_\epsilon)\\
&\quad \text{(by the 3rd axiom of stack theory);}
\end{aligned}
$$

and if t does not contain pop we are finished after the first two steps above since $t' = t_\epsilon$. Observe that t_ϵ (like $t_1, \ldots, t_n$) contains at most $m-1$ pop operators.

Analogously, we define $r_1, \ldots, r_n$ and r_ϵ to which the same reasoning applies. Since, $[\![t]\!]_{RX} = [\![r]\!]_{RX}$ then for any i, $[\![t_i]\!]_{RX} = [\![r_i]\!]_{RX}$ and thus, by induction, $\mathcal{E} \otimes \mathcal{T} \vdash t_i = r_i$. Also, it is easy to see that $[\![t_\epsilon]\!]_{RX}(\epsilon) = [\![r_\epsilon]\!]_{RX}(\epsilon)$ and therefore $\mathcal{E} \otimes \mathcal{T} \vdash t_\epsilon = r_\epsilon$ (the proof is analogous to the proof of the induction base). In summary, we have

$$\mathcal{E} \otimes \mathcal{T} \vdash t = pop(t_1, \ldots, t_n, t_\epsilon) = pop(r_1, \ldots, r_n, r_\epsilon) = r$$

and thereby the proof is completed. □

A.8 Proof of Theorem 6.5

The proof is completely analogous to the proof of Theorem 5.5. We outline the main distinctions. Instead of quasi-real-time deterministic pda we involve nondeterministic push-down quasi-real-time (NPDQRT) machines from [4]. The transition function δ is thus updated as follows:

$$\delta : Q \times (A + \{\epsilon\}) \times \Delta^m \to \mathcal{P}_\omega(Q \times (\Delta^*)^m). \tag{A.9}$$

This function is subject to the condition of being quasi-real-time, i.e. there is a global bound for the lengths of transition chains over machine configurations labelled with internal actions only.

Two acceptance conditions for NPDQRT are possible: (i) by final states and (ii) by empty storage. It is standard to see that a language accepted by empty storage can be accepted by final states. In fact, the construction for ordinary PDAs (see e.g. [**?**]) 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 $(\gamma'_0, \ldots, \gamma'_0)$ (which corresponds to configuration of P with all stacks empty). Clearly, this construction preserves quasi real-timeness.

Conversely, for any m (i) can be modelled by (ii), i.e. a language accepted by final states can be accepted by empty stack: for $m = 1$, we obtain standard push-down automata for which equivalence of (i) and (ii) is well-known [33]; for $m = 2$ this is shown in [**?**]; for any $m > 2$ by [4] 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 when a final state is reached as required by (i).

As shown in [**?**], the languages recognized by NPDQRT with $m = 2$ are properly between context-free and $\mathsf{NTIME}(n)$.

We still need to show that for every m the languages recognized by nondeterministic multistack $\mathbb{T}$-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 5.5, given a nondeterministic multistack $\mathbb{T}$-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

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

The stack alphabet Δ is $\Gamma + \{\Box\}$, the transition function is given as in Theorem 5.5 by changing the number of elements in tuples Q and by allowing for nondeterminism. The acceptance condition is chosen to be by final states, and the final states are:

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

It then follows along the same lines as in the proof of Theorem 5.5 that for any $w \in A^*$, $[\![x_0]\!]_{\mathcal{m}}(w)(\gamma_0, \ldots, \gamma_0) = 1$ iff w is accepted by M with $\langle x_0, \gamma_0, \ldots, \gamma_0\rangle$ 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 (A.9), an initial state $q_0 \in Q$, a set of accepting states $F \subseteq Q$ and an initial stack symbol $\Box$. According to [4], we assume that M is *real-time*, that means that no internal transitions are present.

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

$$\begin{aligned} t^{\mathcal{m}}(q,a)(s_1,\ldots,s_m) &= \emptyset \qquad \text{if } s_i = \epsilon \text{ for some } i,\\ t^{\mathcal{m}}(q,a)(\gamma_1 s_1,\ldots,\gamma_m s_m) &= \{\langle q', s_1' s_1, \ldots, s_m' s_m\rangle \mid \\ &\qquad \langle q', s_1', \ldots s_m'\rangle \in \delta(q,a,\gamma_1,\ldots\gamma_m)\} \text{ if } s_i = \epsilon \text{ for no } i. \end{aligned}$$

A similar argument as in Theorem 5.5 then shows that for any $w \in A^*$, $q \in Q$ and $s \in \Delta^*$ an accepting configuration is reachable from $\langle q, s_1, \ldots, s_m\rangle$ by w iff $[\![q]\!]_{\mathcal{m}}(w)(s_1, \ldots, s_m) = 1$. □

A.9 Proof of Theorem 7.3

Let us denote $\Omega_n = \{(_1,)_1, \ldots, (_n,)_n\}$. We remark that the existence of a homomorphism α from Ω_n to A^* required in the theorem is is still sufficient even in the case of the fixed $n = 2$. To that end, following [**?**], we define a morphism $\gamma : \Omega_n^* \to \Omega_2^*$ sending any $(_n$ to $({}^n_1(_2$ and any $)_n$ to $)^n_1)_2$. As noted in [**?**], $\mathcal{D}_2 = \gamma^{-1}(\mathcal{D}_n)$, which means that if $\gamma(w) \in \Omega_n$ is balanced then w is also balanced. Since the converse is

obvious, the composition $\alpha' = \alpha\gamma : \Omega_n^* \to A^*$ has the property that $\alpha(w) \cdot b_0 \geq b_1$ iff w is balanced whenever the same is true for some $\alpha : \Omega_2^* \to A^*$.

By Chomsky-Schützenberger theorem, $\mathcal{L} = \beta(\mathcal{R} \cap \mathcal{D}_n)$ where $\mathcal{R}$ is a regular language and β is obtained by extending a map $\Omega_n \to A^*$ to a semiring morphism in the canonical way. We use the version of Chomsky-Schützenberger theorem from [**?**] where it is shown that if $\mathcal{L}$ does not contain one-letter words then β can be chosen *non-erasing*, i.e. $\beta(g) = \epsilon$ for no $g \in \Omega_n$. 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

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

Therefore we assume that $\mathcal{L} \cap A = \emptyset$ and β is non-erasing in the remainder of the proof.

As we know from Proposition 4.6 and Definition 5.1, $\mathcal{R}$ can be given by a reactive expression over the boolean semiring $R = B = \{0, 1\}$. We replace in this expression any occurrence of the form $g.\, t$ where $g \in \Omega_n^*$ by

$$a_1.\, \cdots\, a_k.\, \alpha(g) \cdot t$$

where $a_1 \cdots a_k = \beta(g)$ and all the occurrences of $1 \in B$ by b_0. The resulting expression e is a reactive expression for the semiring monad $\mathbb{T}_R$. Note that the assumption that β is nonerasing ensure that e remains guarded. It is then easy to check by definition that

$$[\![e]\!](w) \geq r \cdot b_0 \quad \text{iff} \quad \exists u \in \mathcal{R}.\, r = \alpha(u) \wedge w = \beta(u). \tag{A.10}$$

Suppose, $w \in \mathcal{L} = \beta(\mathcal{R} \cap \mathcal{D}_n)$. Then there is $u \in \mathcal{R}$ which is balanced and such that $w = \beta(u)$. By assumption of the theorem, $\alpha(u) \cdot b_0 \geq b_1$ and by (A.10), $[\![e]\!](w) \geq \alpha(u) \cdot b_0$. Therefore, $[\![e]\!](w) \geq b_1$.

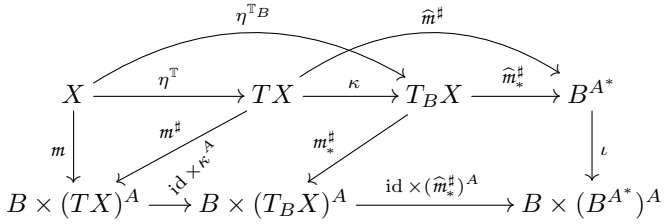
Let us show the converse. Suppose, $[\![e]\!](w) \geq b_1$. Note that $[\![e]\!](w)$ must be representable as a finite sum $\alpha(u_1) \cdot b_0 + \ldots + \alpha(u_k) \cdot b_0$ such that $w = \beta(u_i)$ and $u_i \in \mathcal{R}$ for all i. By assumption of the theorem we obtain $\alpha(u_j) \cdot b_1 \geq b_0$ for some j and therefore u_j is balanced. Since $w = \beta(u_j)$, $u_j \in \mathcal{R}$ and $u_j \in \mathcal{D}_n$ then $u_j \in \mathcal{L}$ and we are done. □

A.10 Proof of Proposition 7.5

The proof is completely analogous to the proof of Theorem 4.8. Let E be the set of $\bullet$-closed algebraic expressions over B_0, R_0. We have defined an L-coalgebra structure on this set given by the functions o and ∂_a. Let Σ be the signature of the algebraic theory associated to the semimodule monad given by the semiring R, i.e. Σ contains a constant $\emptyset$ and binary operator $+$ and unary operators $r \cdot -$ for every $r \in R$. The corresponding syntactic operators according to the grammar in Definition 7.4 turn E into a Σ-algebra such that o and ∂_a (and hence the L-coalgebra structure on E) form Σ-algebra morphisms. The rest of the proof is identical to the proof of Theorem 4.8 with E in lieu of E_{Σ, B_0}. □

A.11 Proof of Theorem 8.1

We will show that the following diagram commutes:



The left-hand triangle commutes by the definition of $m^\sharp$, the right-hand part by the finality of B^{A^*} (recall from (A.6) 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 the finality of B^{A^*} as soon as we establish that the middle part commutes. To see the latter we shall show that all morphisms in this part are $\mathbb{T}$-algebra homomorphisms and that this part commutes when precomposed with $\eta^{\mathbb{T}}_X$. Indeed, the latter follows from the fact that the upper left-hand triangle commutes and since clearly

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

Now to see that all morphisms in the middle part are $\mathbb{T}$-algebra homomorphism, recall first that the monad morphism $\kappa : \mathbb{T} \to \mathbb{T}_B$ induces a functor $\bar{\kappa}$ from the category of $\mathbb{T}_B$-algebras to the category of $\mathbb{T}$-algebras given by $(A,\alpha) \mapsto (A, \alpha\cdot\kappa_A)$. It is easy to see that this functor maps (B, a^{m_*}) to (B, a^m). Now let α and α_*, denote the algebraic structures on $B\times(TX)^A$ and $B\times(TX_B)^A$, which are componentwise given by the structures of the free algebras on X and by a^m and a^{m_*} on B, respectively. Now consider the four morphisms of the middle part of our diagram: (1) $\kappa_X : TX \to T_BX$ is easily seen to be a $\mathbb{T}$-algebra homomorphism from the free algebra $(TX, \mu^{\mathbb{T}}_X)$ to $(T_BX, \mu^{\mathbb{T}_B}_X \cdot \kappa_{T_BX})$ (since κ is a monad morphism) and therefore (2) $\mathrm{id}\times(\kappa_X)^A$ is a homomorphism from $(B\times(TX)^A, \alpha)$ to $(B\times(T_BX)^A, \alpha_*\cdot\kappa_{B\times(T_BX)^A})$; (3) $m^\sharp$ is by definition a $\mathbb{T}$-algebra homomorphism and (4) $m^\sharp_*$ is a $\mathbb{T}_B$-algebra morphism and hence by applying the functor $\bar{\kappa}$ we see it is also a $\mathbb{T}$-algebra homomorphism as desired. □

A.12 Proof of Lemma 8.3

This follows from the fact that B carries an Eilenberg-Moore algebra structure for the countably supported multiset monad M. Equivalently, we have a monad morphism $m : M \to \mathbb{T}_B$ (see [20, Prop. 22.4]). Thus, by forming $(T_BX, \mu^{\mathbb{T}_B}_X \cdot m_{T_BX})$ we obtain an Eilenberg-Moore algebra structure for M on T_BX, i.e., T_BX is an ω-additive monoid. □

A.13 Proof of Theorem 8.6

First, we need a description similar to Lemma A.1 for the observational semantics $[\![-]\!]^\tau_m$. Before we state and prove it we make some auxiliary observations.

Remark A.2. Notice that since $\kappa : \mathbb{T} \to \mathbb{T}_B$ is a monad morphism we have for every $f : X \to TY$ a commutative square

$$\begin{array}{ccc} TX & \xrightarrow{f^\dagger} & TY \\ {\scriptstyle \kappa_X}\downarrow & & \downarrow{\scriptstyle \kappa_Y} \\ T_BX & \xrightarrow[(\kappa_Y \cdot f)^\dagger]{} & T_BY \end{array}$$

which in the do-notation corresponds to the following equation:

$$\kappa_Y\,(\mathsf{do}\ x \leftarrow p; f(x)) = \mathsf{do}\ x \leftarrow \kappa_X(p); \kappa_Y \cdot f(x). \tag{A.11}$$

Remark A.3. In the following we shall need two properties of the ω-additive monoid structure on T_BX.

(1) Sums commute with the Kleisli substitution:

$$\mathsf{do}\ y \leftarrow \sum_{i=1}^{\infty} p_i; f(y) = \sum_{i=1}^{\infty} \mathsf{do}\ y \leftarrow p_i; f(y) \tag{A.12}$$

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

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

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

(2) Similarly, sums commute with the $\mathbb{T}_B$-algebra structure a^{m_*}, i.e. the following equation holds for every countable family of elements $p_i \in T_BB$:

$$a^{m_*}\left(\sum_{i=1}^{\infty} p_i\right) = \sum_{i=1}^{\infty} a^{m_*}(p_i); \tag{A.13}$$

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

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

Lemma A.4. *Given a $\mathbb{T}$-automaton (★), $x_0 \in X$ and $u \in A^*$ then*

$$[\![x_0]\!]^{\tau}_{m}(\epsilon) = o^{m}(x_0) + \sum_{i=1}^{\infty} a^{m} \left(\mathsf{do}\ x_1 \leftarrow t^{m}(x_0, \tau); \ldots; x_i \leftarrow t^{m}(x_{i-1}, \tau); \eta^{\mathbb{T}}_{B} \cdot o^{m}(x_i)\right)$$

$$[\![x_0]\!]^{\tau}_{m}(au) = \sum_{i=1}^{\infty} a^{m} \left(\mathsf{do}\ x_1 \leftarrow t^{m}(x_0, \tau); \ldots; x_i \leftarrow t^{m}(x_{i-1}, a); \eta^{\mathbb{T}}_{B} \cdot [\![x_i]\!]^{\tau}_{m}(u)\right)$$

Proof. We proceed by induction on the argument $w \in A^*$ of $[\![x_0]\!]^{\tau}_{m}$. For $w = \epsilon$ we obtain:

$$\begin{aligned}
[\![x_0]\!]^{\tau}_{m}(\epsilon) &= [\![x_0]\!]_{m_v}(\epsilon) && \text{by definition of } [\![-]\!]^{\tau}_{m} \\
&= o^{m_v}(x_0) && \text{by Lemma A.1} \\
&= o^{m_*}(x_0) + \sum_{i=1}^{\infty} \left(\mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0, \tau); \ldots; t^{m_*}(x_{i-1}, \tau)\right)(o^{m_*}) \\
& && \text{by definition of } o^{m_v} \\
&= o^{m}(x_0) + \sum_{i=1}^{\infty} \kappa_X \left(\mathsf{do}\ x_1 \leftarrow t^{m}(x_0, \tau); \ldots; t^{m}(x_{i-1}, \tau)\right)(o^{m}) \\
& && \text{by repeated application of (A.11) using } o^{m_*} = o^{m} \text{ and} \\
& && t^{m_*} = \kappa_X \cdot t^{m} \\
&= o^{m}(x_0) + \sum_{i=1}^{\infty} (a^{m} \cdot T o^{m}) \left(\mathsf{do}\ x_1 \leftarrow t^{m}(x_0, \tau); \ldots; t^{m}(x_{i-1}, \tau)\right) \\
& && \text{by definition of } \kappa_X \\
&= o^{m}(x_0) + \sum_{i=1}^{\infty} a^{m} \left(\mathsf{do}\ x_1 \leftarrow t^{m}(x_0, \tau); \ldots; x_i \leftarrow t^{m}(x_{i-1}, \tau); \eta^{\mathbb{T}}_{B} \cdot o^{m}(x_i)\right) \\
& && \text{property of do-notation.}
\end{aligned}$$

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

$$
\begin{aligned}
[\![x_0]\!]^{\tau}_{m}(au) &= [\![x_0]\!]_{m_v}(au) && \text{by definition of } [\![-]\!]^{\tau}_{m} \\
&= a^{m_v}\left(\mathsf{do}\ y \leftarrow t^{m_v}(x_0, a); \eta^{\mathbb{T}}_B \cdot [\![y]\!]_{m_v}(u)\right) \\
& && \text{by Lemma A.1} \\
&= a^{m_*}\left(\mathsf{do}\ y \leftarrow \left(\sum_{i=1}^{\infty} \mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0, \tau); \ldots; x_{i-1} \leftarrow t^{m_*}(x_{i-2}, \tau); t^{m_*}(x_{i-1}, a)\right); \eta^{\mathbb{T}_B}_B \cdot [\![y]\!]_{m_v}(u)\right) \\
& && \text{by definition of } t^{m_v} \text{ and since } a^{m_v} = a^{m_*} \\
&= a^{m_*}\left(\sum_{i=1}^{\infty} \mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0, \tau); \ldots; x_{i-1} \leftarrow t^{m_*}(x_{i-2}, \tau); y \leftarrow t^{m_*}(x_{i-1}, a); \eta^{\mathbb{T}_B}_B \cdot [\![y]\!]_{m_v}(u)\right) \\
& && \text{by (A.12)} \\
&= a^{m_*}\left(\sum_{i=1}^{\infty} \kappa_B\left(\mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0, \tau); \ldots; x_{i-1} \leftarrow t^{m_*}(x_{i-2}, \tau); y \leftarrow t^{m_*}(x_{i-1}, a); \eta^{\mathbb{T}}_B \cdot [\![y]\!]_{m_v}(u)\right)\right) \\
& && \text{by (A.11) and since } \kappa \cdot \eta^{\mathbb{T}_B} = \eta^{\mathbb{T}} \\
&= \sum_{i=1}^{\infty} a^{m_*} \cdot \kappa_B\left(\mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0, \tau); \ldots; x_{i-1} \leftarrow t^{m_*}(x_{i-2}, \tau); y \leftarrow t^{m_*}(x_{i-1}, a); \eta^{\mathbb{T}}_B \cdot [\![y]\!]_{m_v}(u)\right) \\
& && \text{by (A.13)} \\
&= \sum_{i=1}^{\infty} a^{m}\left(\mathsf{do}\ x_1 \leftarrow t^{m_*}(x_0, \tau); \ldots; x_{i-1} \leftarrow t^{m_*}(x_{i-2}, \tau); y \leftarrow t^{m_*}(x_{i-1}, a); \eta^{\mathbb{T}}_B \cdot [\![y]\!]_{m_v}(u)\right) \\
& && \text{since } a^{m_*} \cdot \kappa_B = a^{m} \\
&= \sum_{i=1}^{\infty} a^{m}\left(\mathsf{do}\ x_1 \leftarrow t^{m}(x_0, \tau); \ldots; x_i \leftarrow t^{m}(x_{i-1}, a); \eta^{\mathbb{T}}_B \cdot [\![x_i]\!]^{\tau}_{m}(u)\right) \\
& && \text{by definition of } [\![-]\!]^{\tau}_{m} \text{ and renaming } y \text{ to } x_i.
\end{aligned}
$$

□

We are now ready to turn to the proof of Theorem 8.6. To this end we will relate tape automata and a special form of Turing machine called *reactive* Turing machines. Reactive Turing machines were introduced by Baeten et al. [**?**] with the aim to equip TM's with a notion of interaction and so bridge the gap between classical computation and concurrency theory. However, their notion of reactive TM is non-deterministic, and the semantics of reactive TM's is defined in terms of bisimilarity and not in terms of the more classical languages (or traces). So we start by introducing a notion of deterministic reactive TM and its language semantics.

Definition A.5 (Reactive Deterministic Turing Machine (RDTM)). A *deterministic reactive 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 $\square$), 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. So a *configuration* of an RDTM 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 $(q_0, w, (0, \sigma_\square))$ where $w \in A^*$ is the input word and $\sigma_\square$ denotes the constant function on $\square$. Notice that internal transitions leave the remaining input actions untouched while otherwise the head symbol is removed from $w \in A^*$ in a configuration.

Definition A.6 (Language of a RDTM). Let M be an RDTM. 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 A and leads to an accepting state. Notice that due to the internal actions there may be several accepting computations of a word. So an RDTM is only deterministic in the sense that in any configuration there can be no two different moves consuming the same input letter. But an internal transition can happen non-deterministically in any configuration.

That RDTM's are an appropriate model of computations is stated by the following lemma.

Lemma A.7. *The class of languages accepted by RDTM's is the class of semi-decidable languages.*

Proof. We show that ordinary TM's can be simulated by RDTM's and vice versa.

(a) Given an RDTM M it can be simulated by a non-deterministic 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 $\bar{M}$ non-deterministically 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 non-deterministically 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 RDTM is semi-decidable.

(b) Conversely, suppose we have a deterministic TM with input alphabet A. Then M can be simulated by an RDTM $\bar{M}$. 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 on its tape (if any). 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 RDTM. □

Proof (Theorem 8.6). We give for a tape automaton $\mathcal{m}$ as in the statement of the theorem an equivalent RDTM and vice versa.

(a) Given $\mathcal{m}$, we define an RDTM M. For any $x \in X$ and $a \in A$ let $k_{x,a}$ be the minimal natural number as in Definition 3.6 for

$$t^{\mathcal{m}}(x,a) = \langle r, z, t\rangle \in TX.$$

Analogously, l_x be the minimal natural number according to the second clause of Definition 8.5 for

$$o^{\mathcal{m}}(x) : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to 2.$$

The state set of M consists of the states X of $\mathcal{m}$ times a finite memory that can store a finite portion of M's tape and is of the form

$$\{-n, \ldots, 0, \ldots, n\} \times \Gamma^{2n+1}$$

where $n = \max\{l_x, \max_a\{k_{x,a}\}\}$. 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 $j = -n, \ldots, 0, \ldots, n$. The final states of M are those states $x \in X$ together with memory contents $(0, \bar{\sigma})$ that restrict (i, σ) with $o^{\mathcal{m}}(x)(i, \sigma) = 1$; that this is well-defined follows from Definition 3.6. We now describe informally how M simulates $\mathcal{m}$. Since $\mathcal{m}$ can access several symbols from the tape at once we need to simulate transitions of $\mathcal{m}$ 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 $\mathcal{m}$ given by $t^{\mathcal{m}}(x, a)(i, \sigma) = (x', i', \sigma')$ is simulated by the following steps of M (where M starts in state x with the memory contents $(0, \bar{\sigma})$ restricting (i, σ) which is M's tape contents):

(1) M performs a transition that consumes 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 $\ell = -n, \ldots, 0, \ldots, n$ (this is possible by Definition 3.6);
(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 $\mathcal{m}$-transition ends in state x' with a memory content $(0, \bar{\sigma})$ restricting the new tape content (i', σ').

Notice 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 $(0, \sigma_\square)$ iff $[\![x_0]\!]^{\tau}_{\mathcal{m}}(w)(0, \sigma_\square) = 1$. We will prove more generally

that for every state x_0, we have $[\![x_0]\!]^\tau_{\mathcal{M}}(w)(z_0, \sigma_0) = 1$ iff there exists an accepting M-computation from state x_0 starting with tape content (z_0, σ_0).

Before we proceed to the proof recall that the $\mathbb{T}$-algebra structure $a^{\mathcal{M}} : TB \to B$ is given by evaluation. It follows that for every map $f : X \to TB$ the uncurrying of $a^{\mathcal{M}} \cdot f : X \to B \subseteq 2^S$ 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 A.4, $[\![x_0]\!]^\tau_{\mathcal{M}}(\epsilon)(z_0, \sigma_0) = 1$ iff $o^{\mathcal{M}}(x_0)(i_0, \sigma_0) = 1$ or there exists an $i \geq 1$ such that

$$a^{\mathcal{M}}\left(\mathsf{do}\ x_1 \leftarrow t^{\mathcal{M}}(x_0, \tau); \ldots; x_i \leftarrow t^{\mathcal{M}}(x_{i-1}, \tau);\right.$$
$$\left.\eta^{\mathbb{T}}_B \cdot o^{\mathcal{M}}(x_i)\right)(z_0, \sigma_0) = 1.$$

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

$$X \times S \longrightarrow \cdots \longrightarrow X \times S \xrightarrow{o^{\mathcal{M}} \times S} B \times S \subseteq 2^S \times S \xrightarrow{\mathsf{ev}} 2,$$

where the unlabelled arrows form the i-fold composition of the uncurrying of $t^{\mathcal{M}}(-, \tau) : X \to (X \times S)^S$, 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 $t^{\mathcal{M}}(x_k, \tau)(z_k, \sigma_k) = (x_{k+1}, z_{k+1}, \sigma_{k+1})$ for all $0 \leq k < i$, and $o^{\mathcal{M}}(x_i)(z_i, \sigma_i) = 1$. Equivalently, we have an M-computation that performs steps (1)–(3) above i times (simulating τ-steps of $\mathcal{M}$) 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 A.4, we have $[\![x_0]\!]^\tau_{\mathcal{M}}(au)(z_0, \sigma_0) = 1$ iff there exists an $i \geq 1$ such that

$$a^{\mathcal{M}}\left(\mathsf{do}\ x_1 \leftarrow t^{\mathcal{M}}(x_0, \tau); \ldots; x_i \leftarrow t^{\mathcal{M}}(x_{i-1}, a);\right.$$
$$\left.\eta^{\mathbb{T}}_B \cdot [\![x_i]\!]^\tau_{\mathcal{M}}(u)\right)(z_0, \sigma_0) = 1.$$

By a similar argument as in the base case, this is equivalent to the existence of states $x_1, \ldots, x_i$ and tape contents (z_k, σ_k), $1 \leq k \leq i$ such that $t^{\mathcal{M}}(x_k, \tau)(z_k, \sigma_k) = (x_{k+1}, z_{k+1}, \sigma_{k+1})$ for all $0 \leq k < i - 1$, $t^{\mathcal{M}}(x_{i-1}, a)(z_{i-1}, \sigma_{i-1}) = (x_i, z_i, \sigma_i)$ and $[\![x_i]\!]^\tau_{\mathcal{M}}(u)(z_i, \sigma_i) = 1$. 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)–(4) simulating $i - 1$ many τ-steps and one a-step of the given tape automaton $\mathcal{M}$ 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 RDTM $M = (Q, A, \Gamma, \delta, q_0, F)$ we construct an equivalent tape automaton $\mathcal{m}$. We take Q as the set of states and we let

$$o^{\mathcal{m}}(q)(z, \sigma) = 1 \qquad \text{iff} \qquad q \in F$$

and

$$t^{\mathcal{m}}(q, a)(z, \sigma) = (q', z', \sigma'),$$

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 of M, $t^{\mathcal{m}}(q, \tau)$ is defined in the same way.

We need to prove that M accepts a word $w \in A$ iff $[\![q_0]\!]^{\tau}_{\mathcal{m}}(w)(0, \sigma_{\square}) = 1$. More generally, one proves that for every state q_0 and tape content (z_0, σ_0) of M one has $[\![q_0]\!]^{\tau}_{\mathcal{m}}(w)(z_0, \sigma_0) = 1$ 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.

A.14 Tape monad

We show that the tape monad, from Definition 3.6, is well-defined as a submonad of the store monad. Recall that $\sigma =_{i \pm k} \sigma'$ means that $\sigma(j) = \sigma'(j)$ for all j such that $|i - j| \leq k$ and $\sigma =^{i \pm k} \sigma'$ means that $\sigma(j) = \sigma'(j)$ for all j such that $|i - j| > k$. Recall that $\langle r, z, t \rangle : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to X \times \mathbb{Z} \times \Gamma^{\mathbb{Z}}$ belongs to TX iff there exists $k \geq 0$ such that for any $i, j \in \mathbb{Z}$ and any $\sigma, \sigma' : \mathbb{Z} \to \Gamma$ such that $\sigma =_{i \pm k} \sigma'$

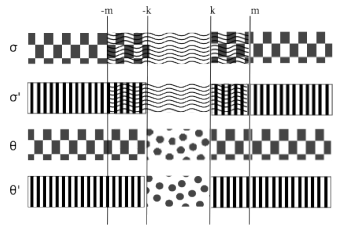
$$\begin{aligned}
&t(i, \sigma) =_{i \pm k} t(i, \sigma') && \text{Ⓐ}\\
&r(i, \sigma) = r(i, \sigma') && \text{Ⓑ}\\
&|z(i, \sigma) - i| \leq k && \text{Ⓒ}\\
&t(i, \sigma) =^{i \pm k} \sigma && \text{Ⓓ}\\
&z(i, \sigma) = z(i, \sigma') && \text{Ⓔ}\\
&t(i, \sigma_{+j}) = t(i + j, \sigma)_{+j} && \text{Ⓕ}\\
&r(i, \sigma_{+j}) = r(i + j, \sigma) && \text{Ⓖ}\\
&z(i, \sigma_{+j}) = z(i + j, \sigma) - j && \text{Ⓗ}
\end{aligned}$$

which we call the *locality conditions* for $\langle r, z, t \rangle$.

To prove that the above definition of submonad is correct, we will use the following property of the relations $=_{i+k}$ and $=^{i+k}$.

Lemma A.8. *Suppose, for some $\sigma, \sigma', \theta, \theta' : \mathbb{Z} \to \Gamma$, $m \geq k$, $\sigma =_{i+m} \sigma'$, $\theta =_{i+k} \theta'$, $\theta =^{i+k} \sigma$ and $\theta' =^{i+k} \sigma'$. Then $\theta =_{i+m} \theta'$.*

Proof. Let us depict the assumptions:



Observe that $\sigma =_{i+m} \sigma'$ and $m \geq k$ imply that $\sigma =_{i+k} \sigma'$, as depicted above (wave pattern).

We want to show that for all j such that $|j - i| \leq m$ we have $\theta(j) = \theta'(j)$. We break this into two cases.

- $|j - i| \leq m \leq k$: $\theta(j) = \theta'(j)$ follows by the assumption $\theta =_{i+k} \theta'$ (polka dots in the picture).
- $k \leq |j - i| \leq m$ and $-m \leq |j - i| \leq -k$: $\theta(j) = \theta'(j)$ follows using following calculation.

$$\begin{aligned} \theta(j) &= \sigma(j) && \text{Chessboard pattern, } \theta =^{i+k} \sigma \\ &= \sigma'(j) && \text{Wave pattern, } \sigma =_{i+m} \sigma' \\ &= \theta'(j) && \text{Vertical stripes pattern, } \theta' =^{i+k} \sigma' \end{aligned}$$

We have to show that the unit satisfies the locality conditions and that (2) for any $p\colon TX$ and $f\colon X \to TY$, $f^\dagger(p) \in TY$.

Recall the definitions of the monad structure:

$$\begin{aligned} \eta(x)(i,\sigma) &= \langle x, i, \sigma \rangle \\ f^\dagger(p)(i,\sigma) &= f(r(i,\sigma))(z(i,\sigma), t(i,\sigma)) \end{aligned}$$

The unit obviously satisfies the locality conditions.

Let $p \in TX$ and denote $p(i,\sigma) = \langle r(i,\sigma), z(i,\sigma), t(i,\sigma) \rangle$. Let $f\colon X \to TY$ and and denote, for every $x \in X$, $f(x) = \langle r_x(i,\sigma), z_x(i,\sigma), t_x(i,\sigma) \rangle$. Let us show that $f^\dagger(p) \in TY$. Suppose, k_p is the parameter of the locality condition for p. For any $\sigma =_{i \pm k_p} \sigma'$ we have that $r(i,\sigma) = r(i,\sigma')$, by Ⓐ, and hence $f(r(i,\sigma)) = f(r(i,\sigma))$. Let k_f be the the parameter for the locality condition for $f(r(i,\sigma))$. Finally let $k = k_p + k_f$ and let us show that this is precisely the parameter needed to prove conditions Ⓐ-Ⓗ for $f^\dagger(p)$.

Let

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

where we fix $x = r(i,\sigma)$, $j = z(i,\sigma)$ and $\theta = t(i,\sigma)$.

Similarly, let

$$f^\dagger(p)(i,\sigma') = f(r(i,\sigma'))(z(i,\sigma'), t(i,\sigma'))$$

$$= \langle r_{x'}(j', \theta'), z_x(j', \theta'), t_x(j', \theta') \rangle$$

where we fix $x' = r(i, \sigma')$, $j' = z(i, \sigma')$ and $\theta' = t(i, \sigma')$. Suppose, $\sigma =_{i \pm k} \sigma'$. Note that this implies that $\sigma =_{i \pm k_p} \sigma'$ and $\sigma =_{i \pm k_f} \sigma'$ and therefore we can assume Ⓐ-Ⓗ for both p (with k_p) and $f(r(i, \sigma)) = f(r(i, \sigma'))$ (with k_f). This immediately gives us, using Ⓑ and Ⓔ for p, that

$$x = x', \qquad j = j'. \tag{A.14}$$

Using the locality conditions Ⓐ and Ⓓ for $f(r(i, \sigma)) = f(r(i, \sigma'))$ we get: $\theta =_{i \pm k_f} \theta'$, $\theta =^{i \pm k_f} \sigma$ and $\theta' =^{i \pm k_f} \sigma'$. Hence, by Lemma A.8, we conclude $\theta =_{i \pm k} \theta'$ and therefore

$$\theta =_{j \pm k_f} \theta', \tag{A.15}$$

which is due to the fact that according to locality condition Ⓒ, $|i - j| \leq k_p$.

Let us now show Ⓐ-Ⓗ for $f^\dagger(p)$.

Ⓐ $t_x(j, \theta) =_{i \pm k} t_{x'}(j', \theta')$
From (A.15) we have $\theta =_{j \pm k_f} \theta'$. Therefore, using the locality conditions for $f(r(i, \sigma))$ we get $t(j, \theta) =_{j \pm k_f} t(j, \theta')$ (by Ⓐ), $t(j, \theta) =^{j \pm k_f} \theta$ and $t(j, \theta') =^{j \pm k_f} \theta'$ (both by Ⓓ). Recall that $|i - j| \leq k_p$. Now since also $\theta =_{i \pm k} \theta'$ (where recall $k = k_p + k_f$) the last two equations imply that $t(j, \theta)$ and $t(j, \theta')$ agree on positions $i - k, \ldots, j - k_f - 1$ and on positions $j + k_f + 1, \ldots, i + k$. Thus, together with $t(j, \theta) =_{j \pm k_f} t(j, \theta')$ we obtain the desired equation.

Ⓑ $r_x(j, \theta') = r_x(j, \theta')$

$$\begin{aligned} r_x(j, \theta) &= r_x(j, \theta') && \text{(A.15) and Ⓑ for } f(x) \\ &= r_{x'}(j', \theta') && \text{(A.14)} \end{aligned}$$

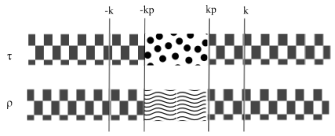
Ⓒ $|z_x(j, \theta) - i| \leq k$

$$\begin{aligned} & |z_x(j, \theta) - i| \\ = \; & |z_x(j, \theta) - z(i, \sigma) + z(i, \sigma) - i| \\ \leq \; & |z_x(j, \theta) - z(i, \sigma)| - |z(i, \sigma) - i| \\ \leq \; & k_f + k_p && \text{Ⓒ for } f(x) \text{ and } p \\ = \; & k. \end{aligned}$$

Ⓓ $t_x(j, \theta) =^{i \pm k} \sigma$

$$\begin{aligned} & t_x(j, \theta) \\ =^{j \pm k_f} \; & \theta && \text{Ⓓ for } f(x) \\ =^{i \pm k_p} \; & \theta \end{aligned}$$

We need to show that for any τ, ρ: $\tau =^{i \pm k_p} \rho \Rightarrow \tau =^{i \pm k} \rho$ and $\tau =^{j \pm k_f} \rho \Rightarrow \tau =^{i \pm k} \rho$. The first equality follows easily because $k_p \leq k$:



For the second one observe that

$$\begin{array}{ll} & |i-m| > k \\ \Leftrightarrow & |i-m| - k_p > k_f \\ \Rightarrow & |i-m| - |j-i| > k_f \;\; j = z(i,\sigma) \text{ and } \text{Ⓒ for } p \\ \Leftrightarrow & |i-m| - |i-j| > k_f \;\; |a| = |-a| \\ \Leftrightarrow & |i-m-i+j| > k_f \quad |a| - |b| \leq |a-b| \\ \Leftrightarrow & |j-m| > k_f \qquad\quad |a| - |b| \leq |a-b| \end{array}$$

This gives us that for all m such that $|i-m| > k$ we have that $|j-m| > k_f$. Therefore, using $\tau =^{j \pm k_f} \rho$ we can conclude that $\tau(m) =_{i \pm k} \rho(m)$.

Ⓔ $z_x(j,\theta') = z_x(j,\theta')$

$$\begin{array}{rl} z_x(j,\theta) = z_x(j,\theta') & \text{(A.15) and Ⓓ for } f(x) \\ = z_{x'}(j',\theta') & \text{(A.14)} \end{array}$$

In the next three items we will use m instead of the index j that appears in the formulation of Ⓕ - Ⓗ above, since we have set $j = z(i,\sigma)$ at the beginning of this proof.

Ⓕ $t_x(j,\theta_m) = t_x(j+m,\theta)_{+m}$, where $\theta = t(i,\sigma)$ as before and $\theta_m = t(i,\sigma_{+m})$.

$$\begin{array}{ll} & t_x(j,\theta_m) \\ = & t_x(j,\theta_{+m}) \qquad \text{Ⓕ for } p\text{: } t(i,\sigma_{+m}) = t(i,\sigma)_{+m}. \\ = & t_x(j+m,\theta)_{+m} \;\; \text{Ⓕ for } f(x). \end{array}$$

Ⓖ $r_x(j,\theta_m) = r_x(j+m,\theta)$

$$\begin{array}{ll} & r_x(j,\theta_m) \\ = & r_x(j,\theta_{+m}) \quad \text{Ⓕ for } p\text{: } t(i,\sigma_{+m}) = t(i,\sigma)_{+m}. \\ = & r_x(j+m,\theta) \;\; \text{Ⓖ for } f(x). \end{array}$$

Ⓗ $z_x(j,\theta_m) = z_x(j+m,\theta) - m$

$$\begin{array}{ll} & z_x(j,\theta_m) \\ = & z_x(j,\theta_{+m}) \qquad \text{Ⓕ for } p\text{: } t(i,\sigma_{+m}) = t(i,\sigma)_{+m}. \\ = & z_x(j+m,\theta) - m \;\; \text{Ⓗ for } f(x). \end{array}$$

□

A.15 $\mathbb{T}$-algebras.

In Definitions 5.4, 6.4 and 8.5, we used $\mathbb{T}$-algebras of the stack monad, nondeterministic stack monad and the tape monad. Let us verify that these were correctly defined. First, we show the following lemma for monads on **Set**.

Lemma A.9. *Let $\mathbb{T}'$ be a submonad of $\mathbb{T}$ and let $\alpha : \mathbb{T} \to \mathbb{P}$ be a monad morphism. Then α restricted to $\mathbb{T}'$ induces a monad morphism $\alpha' : \mathbb{T} \to \mathbb{P}$ such that*

$$\begin{array}{ccc} \mathbb{T}' & \xrightarrow{\alpha'} & \mathbb{P}' \\ {\scriptstyle i}\downarrow & & \downarrow{\scriptstyle j} \\ \mathbb{T} & \xrightarrow{\alpha} & \mathbb{P} \end{array} \tag{A.16}$$

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:

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

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 αi is a monad morphism):

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

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 A.10. *Let $\mathbb{T}_S$ be the store monad over S and let $\mathbb{R}_S$ be the reader monad over S (i.e. $R_S X = X^S$). For any submonad $\mathbb{T}$ of $\mathbb{T}_S$, the monad morphism α sending any $f : S \to X \times S$ to $\pi_1 f : S \to X$ restricts to a submonad $\mathbb{R}$ of $\mathbb{R}_S$.*

The stack monad being a submonad of the store monad over Γ^* induces the submonad $\mathbb{R}$ of the reader monad over Γ^* for which RX consists of those $r : \Gamma^* \to X$ for which there is k such that for any $w \in \Gamma^*$ and any $u \in \Gamma^*$, $r(wu) = r(w)$ whenever $|w| \geq k$. In particular, this makes $B = P2$ in Definition 5.4 an $\mathbb{R}$-algebra and hence a $\mathbb{T}$-algebra.

Analogously, the tape monad (Definition 3.6) induces a submonad $\mathbb{R}$ of the reader monad for which $r : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to X \in RX$ iff there is k such that for all $i, j \in \mathbb{Z}$ and any $\sigma, \sigma' : \mathbb{Z} \to \Gamma$ such that $\sigma =_{i \pm k} \sigma'$ we have

$$r(i, \sigma_{+j}) = r(i + j, \sigma) \quad \text{and} \quad r(i, \sigma) = r(i, \sigma').$$

Therefore $R2$ used in Definition 8.5 forms an $\mathbb{R}$-algebra and hence a $\mathbb{T}$-algebra.

Corollary A.11. *Let $\mathbb{T}_S$ be the nondeterministic store monad over S (i. e. $TX = (\mathcal{P}_\omega(X \times S))^S$) and let $\mathbb{R}_S$ be the nondeterministic reader monad over S (i.e. $R_S X = \mathcal{P}_\omega(X)^S$). For any submonad $\mathbb{T}$ of $\mathbb{T}_S$, the monad morphism α sending any $f : S \to \mathcal{P}_\omega(X \times S)$ to $\mathcal{P}_\omega(\pi_1) f : S \to \mathcal{P}_\omega(X)$ restricts to a submonad of $\mathbb{R}_S$.*

Recall that by Proposition 6.2, the tensor of $\mathcal{P}_\omega$ with m copies of the stack monad over Γ^* is the submonad $\mathbb{T}$ 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 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 $\mathbb{R}$ of the nondeterministic reader monad over $(\Gamma^*)^m$ identified by the condition: $f : (\Gamma^*)^m \to \mathcal{P}_\omega(X) \in TX$ iff 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 $\mathbb{T}$-algebra used in Definition 6.4 is thus obtained by taking $X = 1$.