Towards a Coalgebraic Chomsky Hierarchy
(Short paper)

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

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg
[2] Radboud University Nijmegen

The Chomsky hierarchy plays a prominent role in the foundations of theoretical computer science relating classes of formal languages of primary importance as well as the grammars and machine models that represent those language classes. It is well-known that regular languages can be captured by coalgebras. They form precisely the image of all *finite* coalgebras for the set functor $X \mapsto 2 \times X^A$ under the unique homomorphisms into the final coalgebra, which is carried by all formal languages in this case. There have also been attempts to model context-free grammars (in Greibach normal form) coalgebraically [2, 5] and hence, to capture context-free languages. But what about other language classes and other machine models in the Chomsky Hierarchy?

Here we bring together several ideas in order to treat machine models that are automata extended by extra storage; in particular, we show how to model machines with a stack or a Turing tape coalgebraically, and so we obtain the first coalgebraic account of recursively enumerable languages. Our central notion is that of a $\mathbb{T}$-*automaton*, i.e. a finite coalgebra $m : X \to B \times (TX)^A$, where A is a set of inputs, $\mathbb{T} = (T, \eta, \mu)$ is a finitary monad, which we think as modelling extra storage, and B is an (Eilenberg-Moore) $\mathbb{T}$-algebra $a^m : TB \to B$ of outputs. The semantics of a $\mathbb{T}$-automaton can then be defined via the *generalized power-set construction* [4]: notice first that the functor $LX = B \times X^A$ (being essentially given by product) lifts to the category of $\mathbb{T}$-algebras; then one can extend the coalgebra structure m to the free T-algebra on X to obtain an L-coalgebra $m^\sharp : TX \to B \times (TX)^A$. Observe that the final L-coalgebra is carried by the set B^{A^*} of formal power-series on B. Finally, one defines the semantic map as $[\![\text{–}]\!]_m = h \cdot \eta_X : X \to B^{A^*}$, where $h : TX \to B^{A^*}$ is the unique L-coalgebra homomorphism from $(TX, m^\sharp)$ to the final L-coalgebra.

In order to model machines with pushdown storage we consider the *stack monad*: for a finite set Γ of stack symbols this is the submonad $\mathbb{T}$ of the store monad $(\text{–} \times \Gamma^*)^{\Gamma^*}$ for which elements $\langle r, t\rangle$ of TX satisfy the following restriction: there exists a k (depending on r, t) such that for every $w \in \Gamma^k$ and $u \in \Gamma^*$ we have $r(wu) = r$ and $t(wu) = t(w)u$. Goncharov [1, Proposition 5] has shown that the stack monad can be presented by operations and equations by the *stack theory* w.r.t. $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$: this theory consists of the operations $\mathsf{pop} : n + 1 \to 1$ and $\mathsf{push}_i : 1 \to 1$ for $i = 1, \ldots, n$. These operations are subject to the following axioms:

$$\mathsf{push}_i(\mathsf{pop}(x_1, \ldots, x_n, y)) = x_i, \qquad \mathsf{pop}(\mathsf{push}_1(x), \ldots, \mathsf{push}_n(x), x) = x,$$
$$\mathsf{pop}(x_1, \ldots, x_n, \mathsf{pop}(y_1, \ldots, y_n, z)) = \mathsf{pop}(x_1, \ldots, x_n, z).$$

A *stack* $\mathbb{T}$-*automaton* is a $\mathbb{T}$-automaton for the stack monad for which $B \subseteq 2^{\Gamma^*}$ consists of precisely those predicates p for which there exists a k such that $p(wu) = p(w)$ for all w with $|w| \geq k$, and a^m is given by evaluation: $a^m(r, t)(s) = r(s)(t(s))$.

Theorem 1. *Let m be a stack $\mathbb{T}$-automaton. Then for any $x_0 \in X$ and any $\gamma_0 \in \Gamma$, $\{w \in A^* \mid [\![x_0]\!]_m(w)(\gamma_0)\}$ is a real-time deterministic context-free language. Conversely, any real-time deterministic context-free language can be obtained in this way.*

A similar theorem allows to capture all context-free languages by $\mathbb{T}$-automata. But here one needs non-deterministic machines and so one takes for $\mathbb{T}$ the tensor product of the stack monad with the finite power-set monad $\mathscr{P}_f$.

In order to capture languages beyond context-free ones a treatment of silent transitions is needed (besides a monad $\mathbb{T}$ modelling more powerful storage like e.g. a Turing tape). For this we first consider the *CPS ("continuation passing style")-transform* of a given $\mathbb{T}$-automaton m, which is obtained as follows. First, consider the continuation monad $\mathbb{T}_B$ given by $T_B X = B^{B^X}$. Then observe that having the $\mathbb{T}$-algebra structure $a^m : TB \to B$ is equivalent to having a monad morphism $\kappa : \mathbb{T} \to \mathbb{T}_B$ (see Kelly [3, Prop. 22.4]). We then extend the given $\mathbb{T}$-automaton m to $m_* = (\mathsf{id}_B \times \kappa^A) \cdot m$ with $a^{m_*} = \lambda t.\, t(\mathsf{id}) : T_B B \to B$. One can show that $[\![\text{–}]\!]_{m_*} = [\![\text{–}]\!]_m$. Now assume that B is ω-*additive*, i.e. (besides being a $\mathbb{T}$-algebra) it also is an algebra for the countably supported multiset monad. In other words, B is a commutative monoid with infinite summation. It is not difficult to see that the ω-additive structure extends pointwise to T_B, i.e., for each set X, $T_B X$ is an ω-additive monoid, too. This then allows us to define the elimination of silent transitions from a $\mathbb{T}$-automaton; indeed, suppose we have a $\mathbb{T}$-automaton with input set $A_\tau = A \cup \{\tau\}$, then one can define a $\mathbb{T}_B$-automaton by first forming m_* and then using infinite summation to obtain $m_\tau : X \to B \times (T_B X)^A$ (without τ-transitions)[3]. We define the (τ-free) semantics of m as $[\![\text{–}]\!]^\tau_m = [\![\text{–}]\!]_{m_\tau}$.

Now we are ready to give a coalgebraic description of recursively enumerable languages. For this we consider the *tape monad*: for a finite set Γ of tape symbols this is the submonad $\mathbb{T}$ of the store monad $(\text{–} \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})^{\mathbb{Z} \times \Gamma^{\mathbb{Z}}}$ for which the elements $\langle r, z, t\rangle : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to (X \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})$ of TX satisfy the following restriction: there exists $k \geq 0$ such that for any $i, j \in \mathbb{Z}$ and any $\sigma, \sigma' : \mathbb{Z} \to \Gamma$ if $\sigma =_{i \pm k} \sigma'$[4] then

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

where $\sigma_{+j}(i) = \sigma(i+j)$. A *tape automaton* is a $\mathbb{T}$-automaton for the tape monad with $B \subseteq 2^{\mathbb{Z} \times \Gamma^{\mathbb{Z}}}$ consisting of all those $p \in 2^{\mathbb{Z} \times \Gamma^{\mathbb{Z}}}$ for each of which there exists a k such that $p(i,\sigma) = p(i,\sigma')$ and $p(i,\sigma_{+j}) = p(i+j,\sigma)$ if $\sigma =_{i\pm k} \sigma'$ and with $a^m : TB \to B$ given by evaluation similarly as for stack automata. It can be shown that tape 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 precisely the recursively enumerable-languages.

Theorem 2. *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.*

[3] Unfortunately, space limitations prevent us from giving the technical details here.

[4] We write $\sigma =_{i\pm k} \sigma'$ ($\sigma =_{i\pm k} \sigma'$) if $\sigma(j) = \sigma'(j)$ for all j such that $|i-j| \leq k$ ($|i-j| > k$).

References

[1] Goncharov, S.: Trace semantics via generic observations. In: Heckel, R., Milius, S. (eds.) CALCO 2013. LNCS, vol. 8089 (2013)

[2] Hasuo, I., Jacobs, B.: Context-free languages via coalgebraic trace semantics. In: et al., J.F. (ed.) Proc. Algebra and Coalgebra in Computer Science: First International Conference (CALCO'05). Lecture Notes in Comput. Sci., vol. 3629, pp. 213–231. Springer (2005)

[3] 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)

[4] Silva, A., Bonchi, F., Bonsangue, M.M., Rutten, J.J.M.M.: Generalizing determinization from automata to coalgebras. Log. Methods Comput. Sci 9(1:9) (2013)

[5] Winter, J., Bonsangue, M.M., Rutten, J.J.M.M.: Context-free languages, coalgebraically. In: Proc. Algebra and Coalgebra in Computer Science (CALCO'11). Lecture Notes Comput. Sci., vol. 6859, pp. 359–376. Springer (2011)