Towards understanding Web Scrapers

Loris D'Antoni
PhD Student at University of Pennsylvania
3330 Walnut Street, Philadelphia, PA 19104-6389
lorisdan@seas.upenn.edu
ACM Student Member: 4585012
Advisor: Rajeev Alur

Web Scraping. The last decade has seen a proliferation of programs that operate on data collected from the Internet. In general the database containing such data cannot be accessed directly due to permission restrictions, however these programs can still "scrape" many useful pieces of information directly from HTML pages. A Web Scraper is a program that given an HTML document extracts some data from it and stores it in a table (a relation). An example of such a scraper is a program that given an Amazon search page builds a table containing the names and prices of each product (Fig.1). Even though these transformations are pretty simple, they are often written by non-expert programmers that are not able to provide an efficient implementation. Several tools, like Outwit and Mozenda, have been proposed for helping these users in the task o writing such scrapers, however no clear theory has been developed for supporting them.

A formal model for Web Scraping. In our work we propose a novel model that is able do describe many interesting web scraping tasks while enjoying many decidability and efficiency properties. We build on tree automata and transducers, which have been extensively studied in the context of program analysis [HP03] and XML transformations [MBPS05]. In particular we extend Streaming Tree Transducers [AD12] to output relations instead of trees and to allow symbolic alphabets [VHL$^+$12, DVLM14]. A Streaming Tree Transducers (STT) is an analyzable, executable, and expressive model for transforming unranked ordered trees in a single pass. Given a linear encoding of the input tree, the transducer makes a single left-to-right pass through the input, and computes the output using a finite-state control, a visibly pushdown stack, and a finite number of variables that store output chunks that can be combined to produce the final output. STTs are closed under functional composition, and enjoy decidable equivalence and typechecking.[1]

Since STTs operate on the linear encoding of the input tree they are suitable for defining transformations between HTML documents. However, in the case of Web Scrapers the output is required to be a relation (a table) rather than a tree. To address this issue, we introduce Streaming Tree-to-Relation Transducers (STRT) which extend STTs to output relations. Unlike STTs, STRT's variables are typed and allowed to contain relations. This extension is inspired by the model of symbolic transducers [VHL$^+$12, DVLM14] in which transitions can be labeled with predicates over the input alphabet and each produced output symbol is a function of the input symbol being read. In order to ensure closure and decidability properties, the predicates and functions used by the symbolic transducer should be drawn from a decidable theory that is closed under Boolean operations.

[1]Given two regular languages I and O and a transducers T it is decidable whether for every string $s \in I$, the output $T(i)$ belongs to O. The equivalence problem is only decidable if variables are restricted to be copied in a controlled manner. We defer the reader to [AD12] for further details.

```
...<div id="result_0" name="B00B5S33QE">
   <a href="http://www.amazon.com/..." />
<span class="lrg bold">Fisher-Price McQueen</span>
<ul class="rsltL">
 <li class="med grey mkp2">
  <span class="price bld">38.99</span>
 </li>
</ul></div>...
```

(a) Sample of HTML from Amazon search results.

...	...
Fisher-Price McQueen	38.99
...	...

(b) Output table.

Figure 1: Web scraper extracting product names and prices from Amazon search pages.

Example 1 *Consider the Web Scraper from Fig.1 that extracts the name and price of each product appearing on an Amazon search page.*[2] *This transformation can be captured by an STRT A with a variable n of type string, a variable p of type float, and a variable r of type string $\times$ float. The STRT A reads the HTML document from left to right and whenever it finds a* `div` *for which the* `id` *starts with the prefix* `result` *it performs the following operations: 1) it navigates to the first* `span` *and stores the corresponding value inside the variable n (the product name); 2) it navigates to the first* `span` *of class* `price` *and stores the corresponding value inside the variable p (the price); and 3) it navigates to the closing tag* `/div` *and updates the variable r containing the output relation by adding the new pair (n, p): $r := r \cup \{(n, p)\}$. At the end of the document A outputs the relation r.*

In the following we briefly present the main properties of STRTs

Fast Execution. Given an input tree t, an STRTs can compute the output relation in linear time with a single left-to-right pass over the input t.

Backward Computation. Let A be an STRT that outputs relations of type $D_1 \times \ldots \times D_n$. For every $1 \leq i \leq n$ let φ_i be a predicate over D_i. We define the relation R_o as $\{(a_1, \ldots, a_n) \mid \forall i.\varphi_i(a_i) = \mathit{true}\}$. It is decidable to compute the tree language I such that for every tree $t_1 \in I$, $A(t_1) \subseteq R_o$, and for every $t_2 \notin I$, $A(t_2) \not\subseteq R_o$. This operation can be used to verify a property of the following form: the STRT in Example 1 always outputs product names consisting of alphabetic characters, assuming each Amazon search page satisfies a given HTML format.

Closure under Union, Projection, Selection, and Output reorder. Let A and B be two STRTs that output relations of type $D_1 \times \ldots \times D_n$. 1) The transformation that given an input tree t outputs the relation $A(t) \cup B(t)$ is definable by an STRT. 2) Given an $1 \leq i \leq n$, the transformation that given an input tree t outputs the relation $\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) \mid (a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n) \in A(t)\}$ is definable by an STRT. 3) Given a predicate φ_i over the domain D_i, the transformation that given an input tree t outputs the relation $\{(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in A(t) \wedge \varphi_i(a_i) = \mathit{true}\}$ is definable by an STRT. 4) Given a permutation $i_1 \ldots i_n$ of $\{1, \ldots, n\}$, the transformation that given an input tree t outputs the relation $\{(a_{i_1}, \ldots, a_{i_n}) \mid (a_1, \ldots, a_n) \in A(t)\}$ is definable by an STRT.

Conclusion and future work. We introduced Streaming Tree-to-Relation Transducers (STRT) as an executable, programmable, and analyzable model for Web Scraping. STRTs can be written modularly and then combined using their closure properties. The resulting STRT can be efficiently executed with a left-to-right pass over the input. STRTs can be analyzed, and in particular it can be checked if, for any possible input tree, the output tuples always respect a particular format.

[2] We assume that the input HTML is pre-parsed into tokens and each leaf is a single token of the appropriate type.

References

[AD12] Rajeev Alur and Loris DAntoni. Streaming tree transducers. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, volume 7392 of *Lecture Notes in Computer Science*, pages 42–53. Springer Berlin Heidelberg, 2012.

[DVLM14] Loris D'Antoni, Margus Veanes, Benjamin Livshits, and David Molnar. Fast: A transducer-based language for tree manipulation. In *PLDI'14*, 2014.

[HP03] Haruo Hosoya and Benjamin C. Pierce. Xduce: A statically typed XML processing language. *ACM Trans. Internet Technol.*, 3(2):117–148, May 2003.

[MBPS05] S. Maneth, A. Berlea, T. Perst, and H. Seidl. XML type checking with macro tree transducers. In *PODS'05*, pages 283–294, New York, NY, USA, 2005. ACM.

[VHL+12] Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar, and Nikolaj Bjorner. Symbolic finite state transducers: Algorithms and applications. In *POPL'12*, 2012.