Visualização e animação de autómatos em Ocsigen Framework*

Rita Macedo, Artur Miguel Dias, António Ravara

NOVA LINCS e DI-FCT, Universidade NOVA de Lisboa

Resumo Linguagens Formais e Teoria de Autómatos são bases importantes na formação em Engenharia Informática. O seu carácter rigoroso e formal torna exigente a sua aprendizagem. Um apoio importante à assimilação dos conceitos é a possibilidade de se visualizar interactivamente exemplos concretos destes modelos computacionais, facilitando a compreensão dos mesmos. As ferramentas disponíveis não são completas nem suportam completamente o aspecto interactivo.
Este projecto visa o desenvolvimento de uma ferramenta web interativa, em Português, para ajudar de forma assistida e intuitiva a compreender os conceitos e algoritmos em causa, vendo-os a funcionar passo-a-passo, através de exemplos típicos pré-carregados ou construídos pelo utilizador (um aspecto original da nossa plataforma). A ferramenta deve, por isso, permitir criar e editar autómatos, bem como executar os algoritmos clássicos relevantes, como aceitação de palavras, conversões entre modelos, etc. Pretende-se, também, visualizar não só o processo de construção do autómato, como todos os passos de aplicação de dado algoritmo.
Esta ferramenta usa o *Framework* Ocsigen, pois este proporciona o desenvolvimento de ferramentas web completas e interactivas escritas em OCaml, uma linguagem funcional com um forte sistema de verificação de tipos e, por isso, perfeita para se obter uma página web sem erros. O Ocsigen foi escolhido também porque permite a criação de páginas dinâmicas com sistema de cliente-servidor único.
Este artigo apresenta a primeira fase do desenvolvimento do projecto, sendo já possível criar autómatos, aferir a natureza dos seus estados e verificar passo-a-passo (com *undo*) a aceitação de uma palavra.

Palavras-chave: Linguagens Formais, Autómatos, OCaml, Ocsigen, Ensino, Páginas Web Interactivas.

1 Introdução

Dado o carácter matemático e formal dos tópicos abordados em matérias como linguagens e autómatos, o seu ensino e aprendizagem são exigentes e desafiantes. Vários estudos comprovam estas dificuldades e avaliam a utilidade de

* Trabalho parcialmente suportado pela Fundação Tezos através do projeto FACTOR (`http://www-ctp.di.fct.unl.pt/FACTOR/`) e por fundos nacionais através da FCT – Fundação para a Ciência e a Tecnologia, I.P., no âmbito do NOVA LINCS através do projeto UID/CEC/04516/2019

aplicações para estudar este tema [11, 12, 17, 21]. É importante dar apoio ao trabalho autónomo dos alunos com ferramentas interativas que permitam visualizar exemplos e fazer exercícios. No entanto, a maioria das aplicações são em Inglês e, por terem focos diferentes, nem sempre respondem a todas as necessidades. Enquanto umas são bastante completas, mas por serem *Desktop*, nem sempre estão acessíveis, outras, são de fácil acesso através do *browser*, embora estejam pouco desenvolvidas.

No contexto Português, no que diz respeito aos programas e bibliografias da maioria das disciplinas de Teoria da Computação (ou equivalentes) nas principais Universidades, verifica-se que o material é essencialmente teórico e, que apesar de aquele que é usado em sala de aula poder ser em português, a bibliografia é essencialmente em Inglês. Há, por isso, lugar para uma ferramenta interativa, em Português, que complemente o estudo teórico que já é feito hoje em dia nas aulas.

O objectivo deste projecto é desenvolver uma aplicação, em Português, que facilite o estudo da Teoria da Computação para alunos de Informática, disponível através de um *browser*. A ideia é criar uma ferramenta que possa vir a suportar todos os tópicos dentro da Teoria da Computação, como autómatos finitos deterministas e não deterministas, autómatos de pilha, linguagens regulares, linguagens independentes de contexto, linguagens LL e todas as funcionalidades inerentes as estes tópicos, como conversões, minimizações e testes. Procura-se, ainda, criar uma ferramenta extensível que permita acrescentar funcionalidades, de forma fácil e eficaz. Pretende-se, também, que esta ferramenta esteja, em primeiro lugar, adaptada à disciplina lecionada na FCT-UNL, que venha a incluir exercícios avaliados de forma automática e com *feedback* e que permita, ainda, aos alunos criar os seus próprios exercícios.

Esta plataforma está a ser desenvolvida em *Ocsigen Framework*, ferramenta que permite a criação de sistemas web interativos, totalmente escrito em *OCaml*. Ao tirar partido das características do *OCaml*, é possível obter páginas web completamente funcionais e menos sujeitas a erros. O *Ocsigen* facilita, ainda, a criação de ferramentas web, pois permite escrever cliente e servidor na mesma linguagem, facilitando a programação do sistema.

Neste documento, é apresentada a primeira versão da ferramenta e o seu desenvolvimento. Esta é, para já, uma página simples, mas com algumas funcionalidades importantes: visualização de autómatos, (exemplos disponíveis na aplicação ou criados pelo utilizador), teste de aceitação de palavra (passo a passo ou de forma animada) e verificação da natureza dos estados. Esta ferramenta pode ser acedida em `http://ctp.di.fct.unl.pt/FACTOR/OFLAT` e o código está disponível para consulta em `https://bitbucket.org/rpmacedo/oflat/src/master/`.

2 Trabalho Relacionado

2.1 Ferramentas de visualização existentes

Desde cedo se percebeu que as dificuldades de aprendizagem, e mesmo de ensino, no Estudo das Linguagens Formais e Teoria de Autómatos era um problema recorrente. E que, por ser um tema que gira em torno de processos e máquinas abstractas, a melhor forma de compreender esta problemática seria através de ferramentas pedagógicas. O desenvolvimento destas ferramentas tem sido feito desde o inicio da década de 60 [8]. O artigo *Fifty Years of Automata Simulation: A Review* [8] defende também que de já existirem muitas ferramentas a comunidade cientifica continua a receber novas pois cada uma é diferente, cada uma tem os seus próprios princípios e muitas das vezes novas utilidades. Para além disso cada ferramenta é influenciada pelas ferramentas de desenvolvimento disponíveis no momento.

Existem desde ferramentas textuais, como por exemplo [28], que permite criar e testar autómatos através de texto ou código, sem que estes sejam visíveis graficamente, e ferramentas baseadas na visualização gráfica. Algumas destas serão mais à frente analisadas, com maior profundidade, por terem semelhanças com o tema deste projecto.

No trabalho de pesquisa efectuado, foram encontrados muitos exemplos de aplicações que, de alguma forma, visam colmatar as dificuldades mencionadas pelo recurso a soluções diferenciadas. A título de exemplo, referem-se as seguintes ferramentas: *FSM Simulator* [24] um programa Java que possibilita fazer simulações com autómatos finitos; *Language Emulator* [25] uma ferramenta que permite trabalhar com diferentes conceitos dentro da Teoria de Autómatos e que tem sido usada por estudantes na Universidade de Minas Gerais no Brasil; *jFAST* [15] um software gráfico que permite o estudo de máquinas de estado finitas; *RegeXeX* [27] um sistema interactivo para estudar Expressões Regulares; *Forlan* [23] ferramenta embebida na linguagem Standard ML que permite a experimentação de linguagens formais e autómatos.

É ainda importante referir a ferramenta descrita por *Coffin et. al* em [10] por ser, provavelmente, a primeira a ser desenvolvida na área. Muitas outras são referidas no artigo *Fifty Years of Automata Simulation* [8].

Pode-se falar ainda de bibliotecas desenvolvidas para se Linguagens Formais e Autómatos como Awali [1] (evolução de Vaucanson [9,14] e Vaucanson 2 [13]) e Grail [19],escritas em c++; FAdo, escrita em Python. Awali e FAdo são também e exemplo de bibliotecas que estão a evoluir para aplicações gráficas.

Convém também referir aplicações como *TANGO* [22], *JAWAA* [16], *GUESS* [5] e *VisualAlgo* [4], por terem o objectivo de animar algoritmos ou estruturas de dados, estando relacionados com este projecto devido à sua forte componente interactiva.

Pelo que ficou dito, compreende-se que existem muitas ferramentas que poderiam ser aqui mencionadas. No artigo [20] Secção 2.1 desenvolve-se um pouco mais sobre aquelas que de alguma forma se destacam por diferentes especificidades como JFlap por ser, provavelmente, a ferramenta mais completa disponível;

Automaton simulator por ser uma ferramenta web que permite estudar AF, permitindo verificar a aceitação de frases; FSM simulator, Regular Expression Gym e FSM2Regex por serem também aplicações web e permitirem o estudo de Autómatos Finitos e Expressões Regulares e as suas conversões; Automata Tutor v2.0 por ter um sistema de avaliação e feedback; e AutoMate por ser uma ferramenta com objectivos semelhantes ao deste projecto e que se encontra em desenvolvimento ao mesmo tempo.

Todas as ferramentas mencionadas têm características que, de alguma forma, são comuns ao projecto em desenvolvimento mas não foi encontrada nenhuma que permitisse aos alunos criar os seus próprios exercícios de forma a receber feedback da solução.

2.2 Ambientes de desenvolvimento para a Web

Tipicamente, uma ferramenta web pressupõe dois componentes: cliente e servidor. Estes são normalmente desenvolvidos em linguagens diferentes e, por tal, para que os dados possam ser partilhados entre ambos é também necessário escrevê-los num formato pré-estabelecido.

O lado do cliente é a parte da aplicação web com a qual o utilizador interage, sendo que a cada interação é enviado um pedido ao servidor, que responde com a execução de uma ação ou uma informações da base de dados. Este é maioritariamente desenvolvido pelo recurso a três linguagens: HTML, CSS, *JavaScript*.

O lado do servidor é a parte da aplicação web que indica como esta funciona. O servidor recebe pedidos do cliente e retorna a informação pedida ou a ação que pode ser realizada. O servidor pode ser escrito em diferentes tipos de linguagens como C, C++, C#, PHP, *Python*, *Java* ou até *JavaScript*.

Para que o cliente possa comunicar com o servidor é utilizado um protocolo de comunicação, como HTTP, e a mensagem deve estar num formato convencionado, sendo os mais comuns HTML, XML ou JSON.

Com a necessidade de criação de páginas web mais dinâmicas, e devido ao facto de ser necessária a aprendizagem de muitas linguagens e componentes, começaram a surgir diferentes *frameworks* e bibliotecas com o objectivo de facilitar o trabalho do programador. Alguns visam facilitar o desenvolvimento do design da página, como o *Bootstrap*, cujo o objectivo é criar páginas web *responsive*. Outros pretendem facilitar a criação de aplicações web de uma só página, isto é, reactivas. Temos, como exemplos, *AngularJS*, um framework que estende a linguagem *HTML*; e *React*, uma biblioteca *JavaScript*. De referir ainda *jQuery* uma biblioteca *JavaScript* que pretende simplificar a escrita das *queries* do mesmo e *frameworks* que visam facilitar o desenvolvimento de servidores com muitos acessos à base de dados e que pressupõem reutilização de código, como *Django* e *Ruby on Rails*.

Apesar do surgimento de tantos *frameworks* e bibliotecas, o programador, para criar páginas web, necessita sempre de saber pelo menos *HTML, CSS, JavaScript* (ou um correspondente) e uma linguagem para o servidor.

3 Ocsigen Framework

Ocsigen Framework é uma ferramenta muito completa que se destaca principalmente por permitir a criação de páginas web interativas, escritas totalmente em *OCaml*. Este *Famework* surge da ideia de que a programação funcional é uma solução elegante para alguns problemas de interação nas páginas web [7]. Vem tentar responder aos novos desafios das páginas web, isto é, à necessidade de estas se comportarem cada vez mais como aplicações [6]. Uma das grandes vantagens é compilar o código *OCaml* do cliente para *JavaScript*, o que permite trabalhar em conjunto com esta linguagem e, assim, utilizar um amplo número de bibliotecas, que de outra forma não estariam disponíveis.

3.1 Descrição das componentes

Ocsigen Framework é na realidade um conjunto de diferentes componentes, o que traduz a complexidade desta ferramenta.

Eliom é o componente principal do *Ocsigen*, é uma extensão do *OCaml* para programação sem camadas (*Tierless*) [18]. *Eliom* pretende ser um novo estilo de programação que se enquadra nas necessidades das aplicações web modernas melhor do que as linguagens de programação usuais (desenhadas há muitos anos, para páginas muito mais estáticas [3]). O seu grande objectivo é permitir desenvolver uma aplicação distribuída, totalmente em *OCaml* e como um só programa [3], ou seja, não haver separação entre cliente e servidor. Para que isto seja possível existe uma sintaxe especial para distinguir os dois. O cliente pode aceder facilmente a variáveis do servidor, pois ambos estão escritos na mesma linguagem. Outra importante vantagem resulta do uso da tipificação estática do Ocaml, possibilitando verificar erros e bugs no momento da compilação, havendo a certeza de que se obtêm páginas web corretas, sem problemas nos links ou na comunicação cliente-servidor. Além disso, o *Eliom* resolve automaticamente problemas de segurança frequentes em páginas web. O *Eliom* tem ainda, por base o pressuposto de que se escreve código complexo em poucas linhas. As aplicações desenvolvidas nesta ferramenta correm em qualquer *browser* ou dispositivo móvel, não havendo necessidade de customização.

Js_of_ocaml é o compilador de *OCaml bytecode* para *JavaScript*. É o componente do *Ocsigen* que permite correr a aplicação escrita em *OCaml* em ambientes *JavaScript* como os *browsers* [3] [26]. Possibilita a integração de código *JavaScript* no programa *OCaml*.

Lwt é a biblioteca de *threads* cooperativa para *OCaml* que permite lidar com problemas de concorrência de dados e de *deadlocks*. É a forma standard de construir aplicações concorrentes. Permite fazer pedidos de forma assíncrona, através de promessas. Estas são simplesmente referências que vão ser preenchidas assincronamente, aquando da chegada da resposta.

Tyxml é a biblioteca que permite a construção de documentos HTML estaticamente corretos. Tyxml providencia um conjunto de combinadores, que utilizam o sistema de tipos do OCaml para confirmar a validade do documento gerado.

Ocsigen-start é a biblioteca e *template* de uma aplicação *Eliom* com muitos componentes típicos das páginas web, que visa facilitar a contrução de aplicações web interativas.

Ocsigen-toolkit é a biblioteca de *widgets* que, tal como o *Ocsigen-start*, visa facilitar o desenvolvimento rápido de aplicações web interativas.

4 Animação e visualização de Autómatos

O objectivo deste projecto é a criação de uma ferramenta web que venha a permitir a alunos de informática estudar os vários temas dentro da Teoria de Computação de forma intuitiva e eficaz. Pretende-se que esta ferramenta permita ao aluno trabalhar com Autómatos finitos (minimização, conversão de autómatos não deterministas em deterministas, minimização, conversão em expressões regulares, verificação de aceitação de palavras e geração de palavras de tamanho x), Expressões regulares (simplificação e conversão em AFN), linguagens não regulares, linguagens LL, linguagens independentes de contexto (lema da bombagem e conversão em autómatos de pilha), Autómatos de pilha (conversão em linguagens independentes de contexto) e máquinas de Turing. Para além disso pretende-se ter um sistema de resolução de exercícios com entrega de feedback no qual o aluno pode fazer upload dos seus próprios exercícios.

Ao longo do processo de criação desta versão surgiram alguns desafios que fizeram a utilização do *framework* um pouco complicada. Muito possivelmente devido à mudança de sintaxe do *OCaml* (a sintaxe *camlp4* foi descontinuada para dar lugar a *ppx*), o *Ocsigen* necessitou de ser modificado e nos exemplos de utilização ainda existem algumas inconsistências. Apesar de tudo, no final, conseguiu-se obter uma solução elegante e eficiente que comprova as características do *framework*.

Como este *framework* permite o trabalho conjunto entre *OCaml* e *JavaScript*, para a realização da parte gráfica optou-se pela utilização da biblioteca *JavaScript* de visualização de grafos *Cytoscape.js* [2], por ser muito completa, com várias opções de manipulação e edição dos grafos. Todo o trabalho de verificação ou edição dos grafos é realizado em *Eliom*, sendo a biblioteca utilizada unicamente para representar o grafo, ou seja, é feita uma separação entre o lado lógico e o lado gráfico da aplicação.

Este projecto encontra-se agora em desenvolvimento e já contém algumas funcionalidades importantes: carregar autómatos, gerar autómatos, testar aceitação de palavra e verificar a natureza dos estados. No artigo [20] Secção 4 apresentamos e explicamos as funcionalidades disponíveis da primeira versão da aplicação, que pode ser acedida em `http://ctp.di.fct.unl.pt/FACTOR/OFLAT`. O código completo pode ser consultado em `https://bitbucket.org/rpmacedo/oflat/src/master/`.

5 Conclusões e Trabalhos Futuros

Utilizando o *framework Ocsigen*, está-se a desenvolver uma aplicação web interactiva para apoiar o ensino de Linguagens Formais e Autómatos. A vantagem do *Ocsigen* é permitir desenvolver de forma integrada tando a parte do servidor como do cliente, numa linguagem concisa e segura como o *OCaml*.

Um aspecto central da plataforma é ser extensível; pretende-se incluir mais funcionalidades e outras classes de linguagens, bem como integrar com suporte à avaliação, nomeadamente permitindo a submissão e classificação de exercícios.

Referências

1. Awali. `http://vaucanson-project.org/Awali/index.html`, accessed: 2019-07-09
2. Cytoscape.js. `http://js.cytoscape.org/`, accessed: 2019-06-04
3. Ocsigen - multi-tier programming for web and mobile apps. `https://ocsigen.org/home/intro.html`, accessed: 2019-02-11
4. Visualgo. `https://visualgo.net/en`, accessed: 2019-02-11
5. Adar, E.: Guess: A language and interface for graph exploration. vol. 2, pp. 791–800 (01 2006). https://doi.org/10.1145/1124772.1124889
6. Balat, V.: Ocsigen: Typing web interaction with objective caml. vol. 2006, pp. 84–94 (09 2006). https://doi.org/10.1145/1159876.1159889
7. Balat, V., Vouillon, J., Yakobowski, B.: Experience report: Ocsigen, a web programming framework. vol. 44, pp. 311–316 (09 2009). https://doi.org/10.1145/1596550.1596595
8. Chakraborty, P., C. Saxena, P., Katti, C.: Fifty years of automata simulation: A review. ACM Inroads **2** (12 2011). https://doi.org/10.1145/2038876.2038893
9. Claveirole, T., Lombardy, S., O'Connor, S., Pouchet, L.N., Sakarovitch, J.: Inside vaucanson. pp. 116–128 (01 2005)
10. Coffin, R.W., Goheen, H.E., Stahl, W.R.: Simulation of a turing machine on a digital computer. In: Proceedings of the November 12-14, 1963, Fall Joint Computer Conference. pp. 35–43. AFIPS '63 (Fall), ACM, New York, NY, USA (1963). https://doi.org/10.1145/1463822.1463827, `http://doi.acm.org/10.1145/1463822.1463827`
11. Cogliati, J., W. Goosey, F., T. Grinder, M., A. Pascoe, B., Ross, R., J. Williams, C.: Realizing the promise of visualization in the theory of computing. ACM Journal of Educational Resources in Computing **5**, 1–17 (06 2005). https://doi.org/10.1145/1141904.1141909
12. D'antoni, L., Kini, D., Alur, R., Gulwani, S., Viswanathan, M., Hartmann, B.: How can automatic feedback help students construct automata? ACM Trans. Comput.-Hum. Interact. **22**(2), 9:1–9:24 (Mar 2015). https://doi.org/10.1145/2723163, `http://doi.acm.org/10.1145/2723163`
13. Demaille, A., Duret-Lutz, A., Lombardy, S., Sakarovitch, J.: Implementation concepts in vaucanson 2. vol. 7982, pp. 122–133 (07 2013). https://doi.org/10.1007/978-3-642-39274-0_12
14. Lombardy, S., Poss, R., Régis-Gianas, Y., Sakarovitch, J.: Introducing vaucanson. vol. 2759, pp. 107–134 (06 2003). https://doi.org/10.1007/3-540-45089-0_10
15. M. White, T., Way, T.: jfast: a java finite automata simulator. vol. 38, pp. 384–388 (03 2006). https://doi.org/10.1145/1124706.1121460

16. Pierson, W.C., Rodger, S.H.: Web-based animation of data structures using jawaa. SIGCSE Bull. **30**(1), 267–271 (Mar 1998). https://doi.org/10.1145/274790.274310, http://doi.acm.org/10.1145/274790.274310
17. Pillay, N.: Learning difficulties experienced by students in a course on formal languages and automata theory. SIGCSE Bulletin **41**, 48–52 (01 2009). https://doi.org/10.1145/1709424.1709444
18. Radanne, G., Vouillon, J., Balat, V.: Eliom: A core ml language for tierless web programming. pp. 377–397 (11 2016). https://doi.org/10.1007/978-3-319-47958-3_20
19. Raymond, D., Wood, D.: Grail: A c++ library for automata and expressions. J. Symb. Comput. **17**(4), 341–350 (Apr 1994). https://doi.org/10.1006/jsco.1994.1023, http://dx.doi.org/10.1006/jsco.1994.1023
20. Rita Macedo, Artur Miguel Dias, A.R.: Visualização e animação de autómatos em ocsigen framework (2019), submetido ao ArXiv
21. Sanders, I., Pilkington, C., Van Staden, W.: Errors made by students when designing finite automata (06 2015)
22. Stasko, J.T.: Tango: A framework and system for algorithm animation. SIGCHI Bull. **21**(3), 59–60 (Jan 1990). https://doi.org/10.1145/379088.1046618, http://doi.acm.org/10.1145/379088.1046618
23. Stoughton, A.: Experimenting with formal languages using forlan (01 2008). https://doi.org/10.1145/1411260.1411267
24. T. Grinder, M.: A preliminary empirical evaluation of the effectiveness of a finite state automaton animator. vol. 35, pp. 157–161 (01 2003). https://doi.org/10.1145/611892.611958
25. Vieira, L., Vieira, M., Vieira, N.: Language emulator, a helpful toolkit in the learning process of computer theory. vol. 36, pp. 135–139 (03 2004). https://doi.org/10.1145/971300.971348
26. Vouillon, J., Balat, V.: From Bytecode to JavaScript: the Js_of_ocaml Compiler. Software: Practice and Experience (2013). https://doi.org/10.1002/spe.2187
27. W. Brown, C., A. Hardisty, E.: Regexex: an interactive system providing regular expression exercises. vol. 39, pp. 445–449 (01 2007)
28. Wermelinger, M., Dias, A.: A prolog toolkit for formal languages and automata. ACM SIGCSE Bulletin **37** (09 2005). https://doi.org/10.1145/1067445.1067536