AutoTalk
Talking about Automata with Logtalk

Paul Crocker
Dep. of Computer Science, Covilhã, University of Beira Interior

Março, 2009

Resumo

AutoTalk é um pequeno software para alunos universitários experimentarem com autómatos finitos usando uma linguagem lógica orientada por objectos. O software foi desenvolvido com o objectivo de ser interessante, permitindo a exploração e exposição a várias tecnologias, de ser *razoavelmente* fácil de usar e de poder ser usado em conjunção com outros softwares, em particular o muito conhecido software Jflap com o seu interface gráfica.

Keywords: Automótos finitos. Logic-Programming, Logtalk

1 Introdução

Este software foi pensado para ser usado no desenvolvimento e experimentaçao de autómatos finitos, sejam deterministicos ou não-deterministicos. Embora seja escrito na Linguagem Logtalk [1] (An Open source object-oriented logic programming language) é suficientemente *simples* para ser usado por qualquer pessoa com conhecimentos razoáveis de informática.

Permite a execução de autómatos e a pesquisa sobre as suas propriedades. Vários exemplos são incluídos. Também é incluído um arquivo java do software popular jflap [2] para trabalhar com autómatos, este software representa um autómato através dum ficheiro .jiff, Existem métodos para converter entre os dois formatos.

2 Pre-Requisitos

Logtalk [1] é uma linguagem lógica orientada por objectos. Para o seu uso é necessário um "Back-End"que não é mais nada do que um compilador de Prolog. Muitos compiladores de Prolog existem, tais como SWI, YAP e GNU. Sugiro a utilização dum dos Prologs mais estáveis e bem documentados, nomeadamente o SWI-Prolog[4]. Portanto será necessário ter instalado o compilador de Prolog, SWI Prolog, e o compilador Logtalk. Binários para vários OS's podem ser encontrados nos sites destas ferramentas. Deverá instalar o Swi-Prolog antes de instalar o Logtalk.

3 Instalação

Basta obter o ficheiro zip com o software AutoTalk e descompactar tudo num directório da sua escolha. Incluído no ficheiro zip são os seguintes ficheiros:

- notes.pdf - estes apontamentos.
- script.txt - um ficheiro ilustrando o uso de AutoTalk.
- autotalk.lgt - ficheiro que contêm a definição da Classe principal.

- exemplos.lgt - alguns exemplos de autómatos.
- loader.lgt - ficheiro de inicialização e compilação.
- maquina.lgt - exemplo do livro [3].
- xmljflap.lgt (no sub-directório xmljflap) - um parser de xml.
- JFLAP.jar - o software jflap.
- nfa.jff - exemplo dum autómato jflap.

4 Especificação dum Autómato e os Metódos Disponíveis

4.1 Definição dum Autómato

Um autómato é definido como um objecto da classe *classFa*. Um objecto tem nome, no exemplo em baixo *xyz*, um conjunto de estados iniciais especificado por estadoInicial(nome), um conjunto de estados finais especificado por estadoFinal(nome), um conjunto de transições especificado por delta(estado, símbolo, estado) e um conjunto de transições de string vazia, epsilon(estado1, estado2) como no exemplo em baixo. Pode colocar a definição dum autómato dentro dum ficheiro de texto, neste caso o ficheiro xyz.lgt e compilá-lo dentro do Logtalk com o comando {xyz}.

```
:- object( xyz ,
instantiates(classFa)).

estadoInicial(0).
estadoInicial(ini2).

estadoFinal(2).
estadoFinal(1).

delta(0, a, 1).
delta(ini2, ini2, 1).
delta(1, a, 2).
delta(2, ab, 0).

epsilon(2,1).
epsilon(2,1).

:- end_object.
```

Este autómato tem dois estados inicias (0 e ini2), dois estados finais (1,2) e tem 6 transições, dos quais duas são do tipo "epsilon".

4.2 A Classe Principal

A classe principal *classFa* represente autómatos finitos e contêm todos os métodos para trabalhar com os autómatos definidos como objectos. Métodos (chamados predicados numa linguagem lógica) devem ser definidos como publicos ou privados juntamente com a sua paridade (numero de parâmetros). Os metódos públicos são os métodos de maior interesse e são ilustrados no fragmento de código seguinte. (O código completo pode ser consultado embora seja necessário saber Prolog para a sua compreensão, para os interessados sugiro o livro [3].)

```
:- public(aceita/3).
:- public(aceita/2).
:- public(aceita/1).

:- public(alphabet/1).
```

```
:- public(listInicial/1).
:- public(listFinal/1).
:- public(listStates/1).
:- public(isaDfa/0).

:- public(toJflap/1).
```

4.3 Os Métodos Públicos

A explicação dos métodos públicos dum objecto da classe classFa é dada em baixo:

Tabela 1: default

aceita(Palavra).	Verdade se a palavra é aceita pelo autómato caso contrario Falso.
aceita(Palavra).	Verdade se a palavra é aceita pelo autómato caso contrario Falso.
aceita(dfa, Palavra).	ditto especificando um autómato deterministico
aceita(nfa, Palavra).	ditto especificando um autómato não deterministico
aceita(dfa, Palavra, Trace).	ditto o resultado emite o Trace
aceita(nfa, Palavra, Trace).	ditto o resultado emite o Trace
listInicial(L).	listar os estados inicias
listFinal(L).	listar os estados finais
listStates(L).	listar todos os estados
alphabet(L).	Devolve o alfabeto da autómato
isaDfa.	Verdade se o autómato é deterministico.
toJflap(FileName).	criar ficheiro do autómato no formato .jff para uso com o software jflap.

4.4 Sintaxe e Semântica da Linguagem Logtalk

Um resumo rápido de *tudo* o que precisa de saber:
Uma mensagem é passada a um objecto usando o operador ::. Um Objecto por defeito tem variós métodos pre-definidos, por exemplo, para ver os métodos disponíveis para qualquer objecto pode sempre fazer

```
?- object::current_predicate(L).
```

Para ver todos os objectos actualmente compilados fazer

```
?- current_object(L).
```

Para ver todos os objectos que são da classe classFa

```
?- instantiates_class(L,classFa).
```

5 Utilização - Ambiente Linux

5.1 Arrancar o Logtalk e Carregar o Código e os Exemplos

Deverá navegar para o directório deste código e depois carregar e compilar o software e os exemplos usando os comandos contidos no ficheiro loader.lgt, como é mostrado em baixo.

```
bash shell > swilgt
?- pwd.
/home/users/crocker
true.
?- cd('/home/users/crocker/logtalk/autotalk').
true.
?- {loader}.
```

O loader compilará os ficheiros seguintes:

- autotalk.lgt
- exemplos.lgt
- maquina.lgt
- xmljflap.lgt

6 Exemplos

Dois Autómatos deterministicos e dois não deterministicos são fornecidos para experimentar. Os objectos, adf1 e afd2., são autómatos deterministicos e podem ser usados como no exemplo em baixo.

```
?- afd1::isaDfa.
true.

?- afd1::listStates(S).
S = [1, 2, 0].

?- afd2::isaDfa.
true.

?- afd2::alphabet(L).
L = [b, a].

?- afd2::aceita([a,a,b,b,a,a]).
false.

?- afd2::aceita([a,b,a]).
true.

?- afd2::aceita(dfa,[a,b,a],Trace).
Trace = [0, 1, 0, 1].

?- afd2::aceita(dfa,[a,b,c,a],Trace).
false.

?- afd2::toJflap('afd2.jff').
true.
```

7 Convertendo ficheiros Jflap para AutoTalk

Incluído com este software é o software Jflap (o ficheiro jflap.jar). Um exemplo dum autómato criado usando jflap é dado na figura em baixo e está guardado no ficheiro nfa.jff

Para fazer o "parsing"um ficheiro de jflap (FileName.jff) e criar um ficheiro LogTalk (FileName.lgt) com formato AutoTalk é necessário fazer uso dum XMl parser incluído no Logtlak na secção dos "contribuitions". O predicado convertjflap/2 está incluído no ficheiro xmljflap.lgt e compilado pelo *loader* do AutoTalk. Apenas autómatos simples podem ser convertidos para o formato AutoTalk e

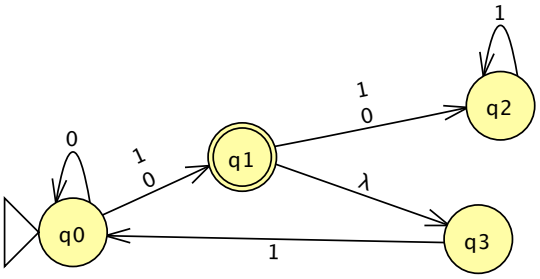

Figura 1: Non-deterministic Finite Automata - Criado com jflap

vice versa. Um exemplo de conversão é dado em baixo. Nota que depois de convertir um ficheiro jflap para AutoTalk será necessário compilá-lo antes de usar.

```
2 ?- convertjflap(autotalk,'nfa.jff').
true.
3 ?- {nfa}.
% <<< loading source file nfa...
% >>> compiling source file nfa...
%     compiling object nfa... compiled
% >>> nfa source file compiled
true.
4 ?- nfa::isaDfa.
false.
```

8 Tips e Truques

8.1 history

Trabalhando com um interpretador de linha de comandos pode ser aborrecido quando estamos sempre a introduzir os mesmos comandos. Sugiro que active o mecansimo de history para ter acesso rapido aos comandos previamente introduzidos. ver "section 2.7 Command-line history"do manual de utilização do swi-prolog [4].

Exemplo. No intreprador de comandos localize o prologo directório de raiz do swi-prolog usando o flag apropriado do predicado current-prolog-flag. Depois saindo do interpretador de Prolog (ctrl-d) abre-se o ficheiro swipl.rc para inserir a linha que especifique o numero de comandos a guardar no histórico dos comandos.

```
1 ? - current_prolog_flag(home,Home).
Home = '/opt/local/lib/swipl-5.6.64'.

ctrl-d
> pico  /opt/local/lib/swipl-5.6.64/swipl.rc
...
:- set_prolog_flag(history, 50).
```

9 Conclusões

Foi apresentado um software para experimentar a utilização de autómatos finitos, deterministicos e não-deterministicos e experimentar a sua utilização num contexto de programação Lógica

A definição dum autómato é muito simples e permite um rapido exploração das suas propriedades. Permite um usuário interessado interagir com o Software Jflap e aproveite o seu interface gráfico, usando um parser de XML para converter entre os dois formatos de representação de Autómatos

Agradecimentos.

Gostava de deixar uma palavra de reconhecimento ao Paulo Moura para o seu trabalho no desenvolvimento do Logtalk, por me ter dado a conhecer o paradigma de programação em lógica e pela sua amizade

Referências

[1] Paulo Moura. Logtalk web site. `http://logtalk.org/`.

[2] Susan Rodger and Thomas Finley. Jflap web site. `http://www.jflap.org/`.

[3] Delfim Torres. *Introdução À Programação em Lógica*. Universidade de Aveiro, 2000.

[4] Jan Wielemaker. SWI-Prolog Home Page. `http://www.swi-prolog.org/`.