I-SCT 2003
Primera Semana de Ciencia y Tecnología
Junio 22-27-2003
Ciudad de Chota - Perú

Autômatos em engenharia de Computação - uma visão unificada

João José Neto

Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia de Computação e Sistemas Digitais
Av. Prof. Luciano Gualberto, trav. 3 n. 158
Cid. Universitária - S. Paulo - SP - Brasil - CEP 05508-900
e-mail: joao.jose@poli.usp.br

Key words: adaptive devices; adaptive automata; unified formalisms.

Abstract
This paper presents adaptive automata, a unified formalism for the representation of a wide range of languages. In this approach, instead of using classic methods for representing formal languages, the proposed formalism operates as a general model, of which the classical simpler models are particular instances. As a desirable consequence, this formalism may be easily explored as a good alternative for many applications. Additionally, the intuitive structure of this model turns it into an excellent option for pedagogical purposes and for building efficient implementations.

Palavras-chave: dispositivos adaptativos; autômatos adaptativos; formalismos unificados.

Resumo
Este artigo apresenta os autômatos adaptativos como um formalismo unificado para a representação de uma ampla gama de linguagens. Com seu uso, ao invés de representar linguagens formais da maneira tradicional, o formalismo proposto opera como um modelo muito geral, a partir do qual os modelos clássicos mais simples se tornam instâncias particulares. Como uma conseqüência desejável, tal formalismo pode ser facilmente explorado como uma excelente alternativa para muitas aplicações. Adicionalmente, a estrutura intuitiva desse modelo o torna uma opção excelente para finalidades pedagógicas e para construir implementações eficientes.

1 Introdução

O presente material relaciona, através de um tratamento sistemático unificado, os pontos da teoria dos autômatos que mais importantes se apresentam sob a ótica do Engenheiro de Computação. Para tanto, diferentemente da maneira clássica de exposição de tais assuntos, usualmente adotada na literatura, será apresentado diretamente, de forma rigorosa, porém intuitiva, um modelo geral, a partir do qual os tradicionais modelos formais, de interesse para a área, serão identificados como casos particulares. Esta maneira de apresentar o assunto permite que se atinjam rapidamente os resultados mais importantes, evitando-se distrações com o estudo de vários modelos formais, desnecessariamente independentes.

Obtém-se com isto uma forma única de representação, que proporciona ao Engenheiro de Computação a oportunidade de elaborar, com base nesta formulação teórica, implementações modulares e estratificadas, cuja complexidade possa ser de tal forma controlada que, em cada caso, sejam estritamente atendidas as exigências particulares pertinentes. Desta maneira, uma vez disponível qualquer implementação, baseada em algum dos casos particulares do modelo, e caso venha a se tornar necessária a inclusão de algum recurso adicional ainda não implementado, bastará que à implementação existente sejam incorporados, modularmente, os recursos faltantes, sem que para isto se exija a realização de todo um novo projeto, e de uma nova implementação, baseada em algum modelo formal diferente do adotado.

O tema tratado é bastante interdisciplinar, e pela sua abrangência e pelo seu caráter fundamental, guarda relação com várias áreas de interesse da Engenharia de Computação, como é o caso, entre outros, da Inteligência Artificial, da Computação Concorrente e da Engenharia de Software, das Interfaces Homem-Máquina, do Ensino por Computador e das aplicações do computador em áreas diversas.

Este trabalho identifica uma forma integrada de tratamento para linguagens textuais, em que são unificados os tradicionais modelos de representação de linguagens, regulares, livres de contexto e dependentes de contexto, com base nas propriedades de tais classes de linguagens, desta forma resultando um modelo geral, que indiferentemente se aplica a todas elas, seja de forma direta ou através de operações simplificadoras de especialização, hierarquicamente aplicáveis.

Não obstante existirem modelos clássicos gerais, como é o caso das Máquinas de Turing, que podem ser aplicados para a definição formal e o tratamento de linguagens de qualquer tipo, o seu emprego direto na prática apresenta severas limitações, a despeito de todas as excelentes qualidades por eles demonstradas no campo teórico, uma vez que processadores de linguagens, diretamente baseados em tais modelos, apresentam-se via de regra como implementações reais de baixíssima eficiência, o que impede seu uso econômico na prática.

As necessidades da área, suscitadas principalmente pela busca de técnicas eficientes de construção de compiladores e outros assuntos correlatos, têm sido satisfeitas, conforme relata a literatura, através de diversos artifícios que, empregados em conjunto com a utilização de autômatos finitos e de pilha, procuram resolver alguns dos problemas que se mostram característicos das linguagens estritamente sensíveis ao contexto.

Isto é feito inicialmente simplificando-se, de forma drástica, as linguagens sensíveis ao contexto, através da sua substituição provisória por linguagens livres de contexto que as contenham, e que se mostrem tais que nelas estejam representados todos os aspectos livres de contexto da linguagem original, da qual tenham portanto sido removidas todas as características dependentes de contexto, e apenas estas.

Dessa maneira, para compensar, em tais casos, as perdas causadas pela simplificação imposta, torna-se necessário re-introduzir o tratamento das características autenticamente dependentes de contexto, e isto é feito, em geral, através da utilização de rotinas semânticas auxiliares, as quais são ativadas quando da

execução das correspondentes transições do autômato finito ou de pilha que implementa a supramencionada aproximação livre de contexto da linguagem.

Embora venha sendo empregado há muito tempo, tal método de tratamento de linguagens dependentes de contexto exibe um vício conceitual relativamente sério, que apresenta reflexos, inclusive, na terminologia usualmente empregada na área: o método trata, através de um enfoque impropriamente dito semântico, a sintaxe dependente de contexto, assunto que nada tem de semântico, mas que, por isso, tornou-se conhecida, no jargão da área, como "semântica estática". Transcrevem-se, a seguir, três importantes trechos da literatura que apresentam argumentos apoiando essa posição.

Inicialmente, [1] afirma:

> *"The terminological boundary between "syntax" and "semantics" is to some degree arbitrary, and different people have drown the distinction in different ways. In this book, we have implicitly taken the position that all those aspects of a programming language which can "reasonably" be explained in terms of symbol sequences (program texts) alone belong to the realm of syntax, and that all other aspects of interest belong to the realms of semantics and pragmatics. The problem now is to clarify what we mean by "reasonably." One useful, but not foolproof, way of looking at the matter is to consider the problem of constructing a compiler for a given subject language. A program expressed in the subject language would be processed in two steps - translation (at "compile time") into machine language by the compiler followed by execution (at "runtime") of the machine code." [...] "The point of view we are taking is that all linguistic aspects, other than code generation and optimization, which are normally handled by the processor at compile time are syntactic in nature and all aspects normally handled at runtime are semantic in nature;" [...] (Pg. 74)*

Em [2] lê-se:

> *"Syntax refers to the ways symbols may be combined to create well-formed sentences (or programs) in the language. Syntax defines the formal relations between the constituents of the language, thereby providing a structural description of the various expressions that make up legal strings in the language. Syntax deals solely with the form and structure of symbols in a language without any consideration given to their meaning." (Pg.1) ..."Though it may be difficult to draw a line accurately between syntax and semantics, we hold that issues normally dealt with form the static text should be called syntax, and those that involve a program's behavior during execution be called semantics. Therefore we consider syntax to have two components: the context-free syntax defined by a BNF specification and the context-sensitive syntax consisting of context conditions or constraints that legal programs must obey." (Pg. 14)*

Na tese [3] é apresentada a questão como tendo origem na terminologia empregada e no ponto de vista adotado para cercar o problema:

> *In order to gain additional power, some means must be used to propagate context-dependent information across the structure of the parse tree. The most successful strategy encountered in the survey seems to be the one done to Knuth (see -§2.2), in which the distribution of context-dependent information is carried out locally within the context-free rules. Knuth's strategy was also applied to adaptable grammars, with favorable results, by Christiansen (§3.3). Therefore, that strategy was identified for the current proposal as another likely ingredient of a successful design. (pg. 66) [...]*
>
> *The notion of programming language development suggests that the entities defined in a program may be thought of as meta-syntactic in nature (in other words, they are statements about the syntax of the language). This view could be treated as an informal principle, but one could also rephrase it more formally as a statement that the domains of semantic values and meta-syntactic values are identical.*
>
> *The proposal takes this unification still further, by requiring that the domain of syntactic values, such as terminal strings, is also identical to the combined semantic/meta-syntactic domain. Besides greatly simplifying the model and enhancing orthogonality, this triple unification also facilitates the technical solution to the objectives for the derivation step relation.*
>
> *The triple unification was foreshadowed by the Christiansen grammar of Example 3.2, in which many of the nonterminals had just two attributes: the inherited language attribute, and a synthesized attribute with either the same name (meta-syntactic) domain, or the (syntactic) domain of terminal strings. A more extensive example [...] also supports the unification. (pg. 68)*

Assim – muitos outros autores importantes também concordam neste polêmico ponto – o enfoque adotado neste trabalho, além de apresentar um formalismo geral, unificado, modular, estratificado, e capaz de suscitar implementações eficientes, pretende contribuir para o resgate do caráter puramente sintático das dependências de contexto, fornecendo mecanismos sintáticos de complexidade relativamente baixa para sua formalização, além de proporcionar formas simples de mapeamento dos mesmos em implementações conceitualmente aderentes, que apresentem um desempenho adequado às aplicações mais importantes e freqüentes da prática.

2 Conceitos principais

Em [4] e em [5] é apresentado o autômato adaptativo, em seus principais conceitos e concepção. Sendo baseados em autômatos de pilha estruturados [6], os autômatos adaptativos herdam daqueles a estrutura hierárquica, o que os tornam muito confortáveis e intuitivos para a representação de mecanismos de aninhamento sintático. Tratando-se, ainda, de um formalismo dotado de recursos de auto-modificação, estes lhe conferem a possibilidade de representar linguagens que não sejam livres de contexto.
Assim sendo, os autômatos adaptativos têm sua operação fundamentada na evolução estrutural espontânea de reconhecedores hierarquicamente estruturados, que efetuam, portanto, o reconhecimento da linguagem que representam como se percorressem uma trajetória no espaço de máquinas de estados, evoluindo de um ponto para outro neste espaço a cada alteração estrutural ocorrida.
A migração entre tais máquinas se dá como conseqüência da execução de transições adaptativas, aquelas que devem promover a auto-modificação do autômato todas as vezes que forem executadas.
Em um autômato adaptativo bem projetado, é possível confinar a ocorrência de transições adaptativas exclusivamente às situações em que o autômato, durante o reconhecimento de uma cadeia de entrada, nesta encontrar indícios da possibilidade de presença de alguma classe de dependência de contexto que ainda não tenha sido tratada pelo autômato.
A operação do autômato deve ser a convencional nos demais casos.
Desta maneira, é possível estratificar a operação do autômato, de acordo com a complexidade da linguagem a que se refere:

- no caso geral, todos os recursos do autômato são utilizados no reconhecimento;
- no caso livre de contexto não-regular, o autômato somente aciona a utilização dos recursos de sub-máquinas e pilha, dispensando os mecanismos adaptativos;
- no caso de linguagens regulares, o autômato pode dispensar também o uso da pilha, operando em essência, portanto, como se fosse um simples autômato finito.

Dessa forma, o formalismo proposto é capaz de aceitar linguagens sensíveis ao contexto sem o auxílio de rotinas semânticas ou de recursos congêneres, lançando mão, para isso, exclusivamente de mecanismos puramente sintáticos.
Mediante extensões ao formalismo proposto, é possível acrescentar-lhe recursos para exprimir mecanismos de transdução sintática, responsáveis por atividades de geração de código, bem como para a representação e tratamento de fenômenos temporais, de sincronização, e probabilísticos, usualmente não cobertos por formalismos similares.
Uma extensão desta natureza foi tema de tese de doutoramento [7], e se refere aos statecharts adaptativos, que incorporam ao formalismo dos statecharts convencionais os recursos adaptativos que lhes permite efetuar dinamicamente, de maneira autônoma, sua própria alteração estrutural.

3 Forma Geral das Transições do Autômato Adaptativo

Um autômato adaptativo pode ser representado por um conjunto de transições da forma:

$$(\gamma\, g\, ,\, e\, ,\, s\, \alpha\,) : A\, ,\, \rightarrow (\gamma\, g'\, ,\, e'\, ,\, s'\, \alpha\,) \, ,\, B$$

onde:

- (γ g , e , s α) representa a situação do autômato antes da aplicação da transição:
  - g representa o conteúdo do topo da pilha antes da aplicação da transição

- e representa o estado do autômato antes da aplicação da transição
- s representa o símbolo da cadeia de entrada a ser consumido pela aplicação

- (γ g' , e' , s' α) representa a situação do autômato depois da aplicação da transição:
  - g' representa o conteúdo do topo da pilha depois da aplicação da transição
  - e' representa o estado do autômato depois da aplicação da transição
  - s' representa o símbolo inserido na cadeia de entrada pela aplicação
- A e B (opcionais) representam ações adaptativas executadas pela aplicação da transição. A aplicação de uma transição que inclua alguma ação adaptativa corresponde à execução de uma transição adaptativa.
  - A representa a ação adaptativa a ser executada antes da transição
  - B representa a ação adaptativa a ser executada depois da transição

Restrições progressivas ao formato das transições acima definidas permitem representar da forma mais simples possível cada uma das diversas classes de linguagens. Conseqüentemente, um único formalismo pode ser usado progressivamente, permitindo o emprego da forma mais simples em cada caso, conforme a necessidade de cada linguagem:

- Para linguagens sensíveis ao contexto, recomenda-se usar as transições na sua forma geral:

$$(\gamma g , e , s \alpha) : A , \rightarrow (\gamma g' , e' , s' \alpha) , B$$

- Para linguagens livres de contexto, pode-se eliminar a representação das ações adaptativas:

$$(\gamma g , e , s \alpha) : \rightarrow (\gamma g' , e' , \alpha)$$

- Para linguagens regulares, pode-se omitir, adicionalmente, a representação da pilha:

$$(e , s \alpha) : \rightarrow (e' , \alpha)$$

3.1 Operação do Autômato Adaptativo

Todo autômato adaptativo pode ser visto como uma máquina de estados que, ao início de sua operação, apresenta uma topologia fixa, predeterminada, denominada máquina de estados inicial E_0.
Na topologia dada por qualquer máquina de estados E_i , o autômato adaptativo opera como um autômato usual, efetuando uma seqüência de transições não-adaptativas, sendo que tal seqüência é finalizada ou pelo término do reconhecimento da sentença, ou então pela execução de alguma transição adaptativa, responsável por alterar sua topologia.
Neste caso, pode-se dizer que o autômato se modifica, assumindo uma nova topologia, representada por alguma outra máquina de estados E_{i+1} , cujo estado inicial de operação deverá ser determinado pela transição executada.
Esta operação se repete até que, na topologia dada pela máquina de estados E_n o reconhecimento se encerre, ou que a cadeia de entrada seja rejeitada pelo autômato. Assim, pode-se dizer que o reconhecimento de uma cadeia $\omega = \alpha_0 \alpha_1 ... \alpha_n$ por um autômato adaptativo pode ser vista como uma trajetória de reconhecimento, dada por

$$\langle E_0 , \alpha_0 \rangle \rightarrow \langle E_1 , \alpha_1 \rangle ... \rightarrow \langle E_n , \alpha_n \rangle$$

O autômato adaptativo inicia sua operação em uma situação inicial dada por (Z_0 , e_0 , ω), ou seja, com a pilha vazia, com a cadeia de entrada completa, e com o autômato, representado pela máquina de estados E_0 , em seu estado inicial e_0 .
Estando o autômato adaptativo, em uma dada ocasião, na situação ($\gamma \pi$, e , $\sigma \alpha$) , é feita uma busca no conjunto de transições do autômato, com o objetivo de localizar alguma transição aplicável a tal situação. Diz-se que uma transição é aplicável quando a situação corrente do autômato não conflitar com o especificado pelas informações que compõem o lado esquerdo da transição (conteúdo do topo da pilha, estado corrente e átomo corrente da cadeia de entrada). Podem ocorrer diversos casos:

- **Existe uma só transição aplicável**. Neste caso a transição correspondente é executada determinística e incondicionalmente
- **Há mais de uma transição aplicável**. Transições internas a uma sub-máquina eliminam outras transições aplicáveis, e, entre estas, transições com consumo de átomos eliminam transições em vazio aplicáveis. Nesta situação, restando apenas uma transição aplicável, esta deve ser executada deterministicamente. Restando mais de uma transição aplicável, constata-se um não-determinismo na operação do autômato, devendo todas as transições correspondentes ser executadas em paralelo.
- **Nenhuma das transições é aplicável**. Caso isto ocorra em um estado não-final, constata-se a rejeição da cadeia pelo autômato, devendo-se considerar que a cadeia não pertence à linguagem definida pelo autômato adaptativo, a não ser que haja, em paralelo, outras tentativas de reconhecimento em curso no autômato (disparados de forma não-determinística). Caso isto ocorra em algum estado final, a cadeia de entrada é aceita se tiver sido completamente consumida e se a pilha estiver vazia, caso contrário será rejeitada se não houver tentativas de reconhecimento em curso no autômato.

As três regras acima são repetidamente aplicadas até que seja atingida uma situação final (γ , e_f , ε), ou então que a cadeia de entrada seja rejeitada pelo autômato.
Descreve-se a seguir a maneira como são utilizadas as transições para promover a operação do autômato adaptativo. Para uma transição aplicável da forma

$$(\gamma\, g , e , s\, \alpha) : A , \rightarrow (\gamma\, g' , e' , s'\, \alpha) , B$$

executam-se, na seqüência abaixo indicada, as atividades seguintes:

- Se A estiver presente, a ação adaptativa correspondente deverá ser executada em primeiro lugar. Se esta ação eliminar a transição correntemente em execução, esta será descontinuada, voltando o autômato adaptativo a buscar, no seu novo conjunto de transições, alguma outra transição que seja aplicável à situação corrente.
- Se g estiver especificado, deverá ser eliminado do topo da pilha
- Se g' estiver especificado, deverá ser empilhado
- Se s estiver especificado, deverá ser consumido da cadeia de entrada, passando a ser o átomo corrente aquele que figurar à sua direita na cadeia de entrada, se existir
- Se s' estiver especificado, deverá ser inserido imediatamente à esquerda do átomo corrente da cadeia de entrada
- o estado e deixa de ser o estado corrente, que passa a ser o estado e'
- Se B estiver presente, a ação adaptativa correspondente deverá ser executada em último lugar.

Para completar esta especificação do funcionamento do autômato adaptativo, resta apresentar a notação com que as ações adaptativas são especificadas, bem como a sua interpretação.
Para que seja possível utilizar uma ação adaptativa, é necessário antes definir a correspondente funções adaptativas F, através de ênuplas cujas componentes são:

- nome F da função
- a lista de parâmetros formais τ_1 , τ_2 , ... , τ_p
- uma lista de variáveis (opcional)
- uma lista de geradores (opcional)
- uma chamada de ação adaptativa anterior (opcional)
- uma lista de ações básicas: de inspeção, eliminação e inclusão de transições
- uma chamada de ação adaptativa posterior (opcional)

As variáveis, os parâmetros e os geradores são representados por nomes simbólicos que representam valores a serem utilizados pelas transições. Ao longo de cada execução da função, esses elementos,

indefinidos ao início dessa execução, não assumem mais que um único valor, o qual, uma vez atribuído, não se altera até o final da execução da função.
Variáveis oferecem uma forma para que se possa referenciar por nome algum valor que não seja constante. As variáveis recebem valores como efeito colateral da execução de ações adaptativas básicas de inspeção ou de eliminação.
Análogos às variáveis, os geradores permitem associar automaticamente valores não repetidos aos nomes que os representam, a cada ocasião em que a função adaptativa for ativada. Tal preenchimento ocorre, também nesse caso, ao início da execução da função adaptativa.
Os parâmetros, associados a valores indefinidos na ocasião do início da execução da função adaptativa, são sempre parâmetros de entrada, e permanecem indefinidos até que se lhes atribua algum valor, o qual será mantido intacto por toda a execução da função adaptativa.
Os parâmetros podem ser preenchidos, também ao início da execução da função adaptativa, por um mecanismo de passagem de parâmetros, o qual atribui a cada um dos parâmetros o valor correntemente associado ao correspondente argumento, indicado na chamada da função.
As ações adaptativas básicas de inspeção, eliminação e inclusão de transições assumem respectivamente as seguintes formas

$$? [(\gamma\, g , e , s\, \alpha) : A , \rightarrow (\gamma\, g' , e' , s'\, \alpha) , B]$$
$$- [(\gamma\, g , e , s\, \alpha) : A , \rightarrow (\gamma\, g' , e' , s'\, \alpha) , B]$$
$$+ [(\gamma\, g , e , s\, \alpha) : A , \rightarrow (\gamma\, g' , e' , s'\, \alpha) , B]$$

onde as expressões entre colchetes representam respectivamente formas de transições a serem pesquisadas, eliminadas ou inseridas no conjunto de transições que define a autômato adaptativo.
No caso das ações básicas de inspeção, o conjunto de transições do autômato adaptativo é pesquisado quanto à ocorrência de transições da forma indicada entre colchetes. Eventuais variáveis, utilizadas para denotar g , e , s , g' , e' , s', bem como nomes e argumentos das ações adaptativas A e B, são em resposta preenchidas com os valores correspondentes, indicados na(s) transição(ões) eventualmente encontrada(s). Caso não seja encontrada nenhuma transição do formato indicado, as correspondentes variáveis permanecerão marcadas como indefinidas. Havendo mais de uma transição nestas condições, constata-se um não-determinismo, e os vários casos possíveis são tratados em paralelo. Ações básicas de inspeção não alteram o conjunto de transições do autômato adaptativo.
Para ações básicas de eliminação, procede-se inicialmente como no caso de ações básicas de inspeção, eliminando-se, do conjunto de transições que na ocasião estejam definindo o autômato adaptativo, todas as transições que tenham sido encontradas como resultado da busca realizada. Se nenhuma transição desta forma for encontrada, nenhuma alteração será executada no conjunto de transições.
Ações básicas de inclusão promovem, no conjunto de transições do autômato, a inserção da transição cuja forma vem indicada entre colchetes. Caso nesta ocasião tal transição já exista no conjunto de transições que define o autômato, nenhuma inserção será efetuada.
Ações adaptativas são incluídas nas transições do autômato na forma seguinte:

$$F (\varphi_1 , \varphi_2 , ... , \varphi_p)$$

onde $\varphi_1 , \varphi_2 , ... , \varphi_p$ são os p argumentos passados à função adaptativa de nome F.
Nas chamadas paramétricas de ações adaptativas, impõem-se as seguintes condições:

- a referência à função adaptativa deve ser feita usando-se o mesmo o nome F com que a função adaptativa foi declarada
- os argumentos $\varphi_1 , \varphi_2 , ... , \varphi_p$ deverão guardar correspondência posicional com os parâmetros formais a que se referem: $\tau_1 , \tau_2 , ... , \tau_p$
- cada um dos argumentos $\varphi_1 , \varphi_2 , ... , \varphi_p$ poderá assumir a forma de qualquer elemento do autômato adaptativo: um nome de função adaptativa, um nome de parâmetro formal, um nome de variável, um nome de gerador, um símbolo de pilha, um símbolo do alfabeto de entrada, um nome de estado.

As ações adaptativas anterior e posterior são normalmente representadas por meio de chamadas de funções adaptativas, em geral paramétricas.

4 Histórico, Evolução e Estado da Pesquisa

O modelo formal apresentado constitui o principal produto de uma busca, de muitos anos, de um formalismo para a representação de linguagens, que se mostrasse ao mesmo tempo poderoso, conceitualmente simples, eficiente em sua operação, fácil de ser implementado, e que apresentasse um forte potencial para aplicações complexas diversificadas.
O modelo deveria ser ainda suficientemente abrangente para que pudesse representar um passo em direção à formalização econômica de linguagens dependentes de contexto, não diretamente tratáveis através das eficientes e consagradas técnicas populares que são usualmente empregadas na construção de compiladores.
Por outro lado, apesar de existirem muitas técnicas e modelos específicos voltados para a análise e processamento de linguagens dependentes de contexto, tais técnicas usualmente exibem uma complexidade significativa, motivando a busca de alternativas mais eficientes e menos complexas.
Outra consideração importante, já mencionada anteriormente, é que os formalismos usualmente empregados para esta finalidade procuram recair na utilização de tratamentos livres de contexto, aplicados sobre formas simplificadas de tais linguagens, exigindo desta forma que sejam empregadas operações complementares adicionais, destinadas a cobrir os aspectos da linguagem não capturados por tal formalismo simplificado, livre de contexto, o que acarreta uma impropriedade conceitual, já que tais aspectos são de caráter genuinamente sintático.
Um resultado extremamente relevante deste estudo é o de que o formalismo proposto pelos autômatos adaptativos, pela sua generalidade e estrutura estratificada, oferece uma forma declarativa, unificada e hierárquica de representação para a definição de todos os tipos de linguagens textuais, proporcionando, em adição, um substrato conceitual que permite implementações muito eficientes.
Outra propriedade digna de menção, apresentada pelos autômatos adaptativos, é a sua intuitiva simplicidade estrutural, que lhe concede a vantagem de servir como ferramenta didática, bastante apropriada para o ensino de aspectos de implementação de linguagens a alunos de cursos que não sejam obrigatoriamente tão especializados e ricos em teoria como os cursos de Ciências da Computação ou de especialidades afins.
Uma característica adicional, de alta relevância, apresentada pelos autômatos adaptativos, corresponde à sua adequação como fundamento para a criação de ferramentas de geração automática ou quase automática de processadores para linguagens textuais, possivelmente de compiladores ou de interpretadores para tais linguagens.
Ainda decorrente das características dos autômatos adaptativos, a propriedade mais importante por eles exibida talvez seja a possibilidade de se auto-modificarem, de forma controlada, conforme regras estabelecidas como parte do seu próprio formalismo.
Isto faz dos autômatos adaptativos um modelo capaz de representar conhecimento, incorporar informações e realizar atividades básicas de aprendizagem, o que os torna uma interessante forma alternativa de fundamento para muitas atividades da área da Inteligência Artificial.
Conforme foi anteriormente mencionado, a presente linha de pesquisa tem apresentado um permanente progresso, obtido passo a passo através de uma sucessiva busca de versões de modelos formais adequados à conceituação e à realização eficaz de linguagens, particularmente das sensíveis ao contexto, através do emprego de formalismos poderosos, com elevado potencial de aplicação prática, e que podem ser explorados de forma tal que nunca façam uso de recursos com maior complexidade conceitual do que aquela que for estritamente necessária em cada situação.

Inspirado originalmente no funcionamento das gramáticas W de dois níveis, e inicialmente proposto como uma espécie de autômato também de dois níveis, a concepção do autômato adaptativo ganhou corpo e se aproximou da forma atual ao ser idealizado o conceito de função adaptativa.
Em sua proposta mais antiga, uma função adaptativa exibia caráter procedimental, e se assemelhava muito às ações semânticas usualmente em muitas ferramentas que utilizam gramáticas de atributos ou formalismos similares, para a construção semi-automática de compiladores.
A conscientização acerca da maior adequação de especificações declarativas conduziu à versão atual do modelo, na qual as operações executadas pelas ações adaptativas não têm a sua seqüência estabelecida pela ordem física em que são especificadas tais operações, mas pelas convenções do próprio funcionamento do autômato, sendo consideradas não mais como seqüências, mas como conjuntos de operações, a serem dinamicamente ordenadas de acordo com regras implícitas claras, impostas pela própria definição da operação do autômato adaptativo.
Em diversas ocasiões o novo modelo foi submetido a críticas, tendo sido amplamente discutido por especialistas, profissionais e muitos estudantes, em eventos, cursos de Graduação e de Pós-Graduação.
Como publicação prévia, diversos artigos foram gerados e apresentados em diversos fóruns, incluindo congressos e palestras proferidas na Escola Politécnica, no Instituto de Ciências Matemáticas de São Carlos, no Instituto Tecnológico de Aeronáutica, na Universidade Estadual de Campinas e no Instituto de Matemática e Estatística da Universidade de São Paulo, com a finalidade de divulgar os modelos emergentes, e discutir de forma mais ampla os conceitos a eles associados.
Inicialmente tratava-se de uma versão ainda não-adaptativa, que se apresentava na forma de autômato de pilha estruturado, o qual, desde então, vem sendo extensivamente utilizado no ensino de disciplinas relacionadas com linguagens formais e autômatos, linguagens de programação e compiladores, tendo sido utilizado pelo autor como base para a elaboração de um livro didático de introdução à compilação [6], atualmente utilizado em disciplinas de compilação em diversas instituições.
Em particular, de um trabalho de pesquisa escolar, realizada por um aluno de final de curso de Graduação em engenharia, resultou um texto publicado nos Anais da Escola Politécnica da Universidade de São Paulo, na forma de matéria voltada à apresentação de uma proposta de automatização para a tarefa da geração de reconhecedores baseados em autômatos de pilha estruturados a partir de gramáticas livres de contexto [8].
Esse detalhado relatório técnico constituiu a primeira documentação de uma experiência realizada na elaboração de um gerador de autômatos de pilha estruturados a partir de gramáticas livres de contexto.
Desenvolvimentos subseqüentes culminaram com a elaboração da versão adaptativa do modelo, motivados pela necessidade de uma boa ferramenta teórica para a representação de dependências de contexto e de extensibilidade.
Esta pesquisa teve como principal fruto a tese de livre-docência do autor, com desdobramento em diversos seminários proferidos na Escola Politécnica, e uma primeira publicação internacional na ACM SIGPLAN Notices [5].
Com o formalismo assim proposto, surgiu a necessidade de validá-lo como modelo de computação, o que suscitou diversos trabalhos de mestrado e de doutorado, cujos objetivos principais foram demonstrar, de maneira rigorosa, as principais propriedades do autômato adaptativo, como modelo formal de computação e como forma de representação de linguagens.
Como principal resultado, foi demonstrada a equivalência entre os autômatos adaptativos e a máquina de Turing, e como sub-produto, apresentaram-se, de forma concreta, os mapeamentos de diversas construções das linguagens de programação usuais para a forma de autômatos adaptativos, constatando assim sua adequação para essa finalidade.
Em outra frente complementar, alguns trabalhos de mestrado e de doutorado, confirmaram a praticidade e a utilidade do autômato adaptativo como forma alternativa de representação de conhecimento, tendo sido usado na representação e no processamento de diversos aspectos das linguagens naturais, tendo-se dessa forma demonstrado a capacidade que têm os autômatos adaptativos

de representarem o conhecimento de forma análoga à encontrada em formalismos consagrados dessa área, como é o caso de formulações, tais como ATN, DCG, Frames, XG, RTN e muitos outros [9].
Estão concluídas ou em curso diversas pesquisas ou desenvolvimentos ligados aos mecanismos adaptativos, suscitados pelo assunto aqui mencionado. Dentre elas, as mais significativas são apresentadas e comentadas a seguir.

- Uma *ferramenta para desenvolvimento de autômatos adaptativos* – a necessidade de ferramentas para o auxílio ao desenvolvimento de autômatos adaptativos e suas aplicações inspirou um trabalho de mestrado que produziu como resultado um ambiente para a especificação e simulação de um subconjunto considerável dos recursos dos autômatos adaptativos, e que foi utilizado e aprimorado para dar suporte à construção de processadores de linguagens de programação [10]. Mais tarde, com os avanços na formulação e com a correspondente ampliação da abrangência dos formalismos adaptativos, optou-se criar um novo e mais moderno ambiente, que fosse mais flexível ainda, e voltado a aplicações mais gerais. Parte de um trabalho de doutorado, esse novo programa é capaz de aceitar especificações formais de autômatos adaptativos, efetuando a partir delas a simulação da operação de tais autômatos. Este programa oferece uma interface gráfica com a qual é possível acompanhar o projeto e efetuar com facilidade a depuração do autômato desejado. Esta ferramenta tem como principais aplicações a disponibilização de uma plataforma de ensaio para autômatos em geral, em que os aspectos da teoria podem ser experimentados, e que está sendo utilizada para facilitar e mecanizar o desenvolvimento de linguagens dependentes de contexto e de seus processadores. Um outro recurso incluído nesta ferramenta permite explorar o modelo ensaiado como sendo um transdutor adaptativo, cujas saídas podem constituir um código-objeto para a linguagem de entrada [10]. Esse trabalho, materializado na ferramenta Adaptools [11] está implementado e em operação, e está integralmente disponibilizado na forma de um software livre [12].
- Um *formalismo* e uma *ferramenta para o desenvolvimento de statecharts adaptativos* - tema de uma pesquisa de doutorado concluída, este projeto teve como meta a criação de um modelo voltado à representação de sistemas concorrentes e de tempo real, fundamentado nos statecharts propostos por Harel [13], estendidos com o conceito de adaptatividade inspirado no funcionamento dos autômatos adaptativos. Como resultado, obteve-se o modelo dos statecharts adaptativos, para os quais foi implementado um protótipo de ferramenta de desenvolvimento, através da qual se pode modelar, testar e depurar statecharts adaptativos, cuja operação pode ser acompanhada visualmente através de animação gráfica. Em prosseguimento a essa pesquisa, uma pesquisa adicional culminou numa dissertação de mestrado, em que se apresentaram os *statecharts adaptativos sincronizados* uma extensão do formalismo e da ferramenta acima mencionados de modo que incorporem o conceito de tempo, na forma de mecanismos de sincronização explicitados com o auxílio de redes de Petri [14].
- Como produto de uma pesquisa associada a outro programa de doutoramento [15], um estudo mais profundo da teoria dos autômatos adaptativos, visando ao estabelecimento da relação que é guardada entre os autômatos adaptativos e outros formalismos importantes, em particular com as máquinas de Turing e com as gramáticas modificáveis, as primeiras devido a seu poder de representação e pelo respeito a que fazem jus como modelo geral de computação, e as últimas devido à similaridade estrutural e conceitual como formalismo adaptativo em sua forma gramatical. Os principais resultados intermediários obtidos foram a demonstração da equivalência entre a Máquina de Turing e os Autômatos Adaptativos, e algumas propostas de formalismos gramaticais adaptativos, em particular inspiradas nas gramáticas adaptáveis recursivas propostas pos Shutt [3]. Como produto paralelo desta pesquisa pretende-se propor uma metodologia de conversão automática do formalismo gramatical adaptativo para o do autômato adaptativo equivalente. Conta-se, para tanto, com um algoritmo já consagrado, que efetua tarefa semelhante

para o caso não-adaptativo, e que deverá ser usado como base para extensão, da qual resultará o novo algoritmo desejado.

- Um curioso e significativo produto dessa tecnologia pode ser experimentado através da utilização de um programa baseado em formalismos denominados Redes de Markov Adaptativas, que foram empregadas como substrato do programa Lassus [16], um fisicamente pequeno e estruturalmente simples, porém conceitualmente sofisticadíssimo sistema de geração automática de música por computador, dirigido por regras configuráveis, o qual é capaz de gerar e de tocar, no computador, de forma autônoma, música de qualidade surpreendente. Esse programa está disponível em [11].
- Estão em curso algumas pesquisas adicionais, a serem publicadas em breve, envolvendo a teoria dos autômatos e de outros formalismos adaptativos. Delas tem emanado, nos últimos anos, uma série de resultados mais avançados, envolvendo a criação e aplicação de novos dispositivos adaptativos, bem como muitos interessantes avanços na teoria a eles referente, e inúmeras aplicações em diversas áreas do conhecimento, tais como a Inteligência Artificial, a Engenharia de Software, a Educação auxiliada por Computador, a Especificação e Implementação de Linguagens de Programação, a Otimização de Código em compiladores, a Inferência automática, o Processamento de Linguagens Naturais, o Auxílio computadorizado à tomada de decisões, entre tantas outras.

Por falta de espaço, não estão aqui relacionados todos os trabalhos já desenvolvidos, mas muitas informações sobre outros estudos realizados, outros formalismos já desenvolvidos e outros softwares disponibilizados podem ser encontradas em [21] e em [11].

5 Trabalhos correlatos

Os trabalhos clássicos relativos ao tratamento de linguagens dependentes de contexto costumam dar ao tema enfoques teóricos, centrados nas máquinas de Turing. Na mesma direção desta pesquisa, que culminou com a criação do autômato adaptativo, encontram-se diversos trabalhos em que este foi inspirado. Em particular, os trabalhos de Conway [17] e de Lomet [18], sugerindo autômatos de pilha baseados em máquinas de estado separáveis e mutuamente recursivas, e as gramáticas de Lindenmayer, que foram utilizadas por seu autor na simulação do crescimento de algas filamentosas, forneceram os princípios para a criação do modelo do autômato de pilha estruturado, versão não-adaptativa do formalismo desenvolvido na presente pesquisa.

Mais recentemente, formulações diversas passaram a preocupar-se em dar embasamento formal a aspectos da especificação de linguagens de programação dependentes de contexto.

Sobressaíram-se, nesta classe de formalismos, as gramáticas W, propostas por Van Wijngaarden, as gramáticas de atributos, o formalismo VDL (Vienna Definition Language) e outros, menos destacados.

Há poucos anos, com o crescimento da importância da inteligência artificial, e, em particular, do interesse pelo processamento automático de linguagens naturais, surgiram as gramáticas de cláusulas diretas (DCG), as gramáticas de extraposição (XG), as redes de transições recursivas (RTN), as redes de transições aumentadas (ATN), as gramáticas auto-modificáveis e as gramáticas evolutivas, estas duas últimas filosoficamente muito aderentes ao formalismo dos autômatos adaptativos.

O surgimento simultâneo e independente das gramáticas evolutivas e dos autômatos adaptativos trouxe para esta área de pesquisa uma nova visão, mais pragmática, da formalização de linguagens dependentes de contexto, oferecendo mecanismos práticos para a formulação de tais linguagens e para a obtenção rápida e simples de autômatos eficientes a partir de tais gramáticas.

Muito recentemente, trabalhos paralelos na mesma linha surgiram, dando oportunidade para que as idéias de utilização de mecanismos adaptativos se generalizassem ainda mais. Pode-se apontar, nesta direção, os trabalhos de Rubinstein e Shutt [19,20], elaborado na Universidade de Worcester, que desenvolve os Autômatos Finitos Auto-modificáveis, um formalismo muito similar ao dos Autômatos Adaptativos.

Para esta modelagem formal, Shutt aponta uma série de fatos que atestam a eficácia dos mecanismos adaptativos para a formalização de linguagens sensíveis ao contexto, e discute detalhadamente a abrangência lingüística daqueles.
Não se deve deixar de mencionar trabalhos correlatos, porém voltados a formalismos gramaticais, dedicados ao estudo da representação de linguagens através de mecanismos formais diferentes dos autômatos. Esses guardam uma similaridade bastante grande com as Gramáticas Adaptativas apresentadas em [15].
Assim, Christiansen faz um excelente relato do estado da pesquisa na área das gramáticas adaptáveis, através de uma compilação de um conjunto expressivo dos trabalhos publicados sobre o assunto até fins de 1990. Shutt, em sua tese, apresenta também um relato organizado sobre este tema, como base para situar o seu próprio trabalho dentro da área.

6 Vantagens das inovações introduzidas

As características introduzidas pelo modelo do autômato adaptativo trouxeram à área algumas contribuições significativas, dentre as quais se distinguem as que vêm discutidas a seguir.
Resgate do caráter sintático das dependências de contexto – esta vantagem destaca-se das demais uma vez que por ela é possível eliminar, se assim for conveniente, a figura da semântica estática na caracterização das linguagens de programação. Isto é possível erradicando-se as chamadas de rotinas auxiliares externas ao autômato, usualmente conhecidas como *ações semânticas*, empregadas no tratamento das linguagens quanto às dependências de contexto que apresentam. Em lugar de programas auxiliares, alheios ao formalismo do autômato, os autômatos adaptativos incorporam recursos sintáticos que lhe permitem a retenção e manipulação, em sua própria estrutura topológica, de informações acerca do programa, dispensando assim o uso explícito de tabelas, variáveis etc.
Além do poder de computação que apresentam, equivalente à máquina de Turing, tais recursos permitem que sejam empregados confortavelmente autômatos adaptativos na realização de todas as atividades de compilação, por meio de um tratamento estritamente sintático – naturalmente, em nem todos os casos se mostra vantajoso modificar a metodologia, mas essa é uma possibilidade concreta:

- eliminação de tabelas de símbolos (nomes, palavras reservadas) e de atributos – as informações usualmente contidas em tais estruturas de dados passam a ser representadas como parte da própria topologia do autômato.
- eliminação da distinção e separação entre os tratamentos léxico, sintático e semântico em uma linguagem – sem impedir a utilização desta estratificação clássica, o formalismo adaptativo permite que a linguagem representada seja considerada pelo autômato de uma forma integrada, o que proporciona a possibilidade de uma formalização uniforme, inclusive nos complexos casos das linguagens sintaticamente extensíveis.
- tratamento sintático dos escopos dos nomes – incorporado à análise léxico-sintática, o tratamento de escopos para os nomes pode passar a ser efetuado pela manipulação da estrutura topológica do autômato, criando-se ou bloqueando-se acessos às partes do mesmo que representam elementos da linguagem sujeitos a regras de escopo, de acordo com a visibilidade contextual de tais elementos ao longo da análise.
- tratamento sintático dos atributos associados aos nomes – operações de verificação da validade do uso de elementos da linguagem ao longo da análise, em particular as verificações de tipo (“type-checking”) podem ser facilmente implementadas, nos autômatos adaptativos, de forma similar ao que se mencionou no caso dos escopos: o emprego de caminhos, dinamicamente variáveis, de acesso aos elementos da linguagem sujeitos a atributos impostos pela estrutura do texto-fonte, pode fazer com que o autômato adaptativo efetue a aceitação exclusiva daqueles elementos que se

mostrem compatíveis com os atributos que lhes tenham sido associados, previamente, em partes já analisadas do texto-fonte.

- tratamento sintático da declaração e da chamada de procedimentos – pode-se considerar este como um caso particular do tratamento de atributos, no qual a declaração de um nome, como sendo o nome de um procedimento, impõe que sua subseqüente utilização seja permitida apenas em situações particulares, como costuma ocorrer em comandos de chamada de procedimento, os quais usualmente exibem sintaxe própria e fortemente dependente dos demais atributos associados a tais nomes, exigindo que seja incorporado um mecanismo de verificação sintática, responsável por garantir a aderência entre os atributos associados aos argumentos, e os seus correspondentes, referentes aos parâmetros formais do procedimento.

Autômatos adaptativos incluem, portanto, recursos conceituais suficientes para dar ao seu usuário a opção de dispensar estruturas explícitas de dados, usualmente empregados nos métodos convencionais de análise e de compilação de linguagens de programação usuais.
Em contrapartida, devem ser consideradas, na prática, as inconvenientes e inevitáveis alterações nas dimensões do autômato, resultantes das ampliações físicas sofridas pelo mesmo como decorrência da execução de tais atividades adaptativas.
No extremo, os autômatos adaptativos permitiriam eliminar até mesmo sua pilha sintática explícita, cuja função pode também ser substituída por meio da incorporação, ao autômato, de alterações topológicas adequadas, realizadas dinamicamente por intermédio de ações adaptativas.
Do ponto de vista pragmático, entretanto, a presença de uma pilha explícita como parte do autômato adaptativo evita a proliferação exagerada de seus estados e transições como decorrência até mesmo de atividades triviais, como é o caso no tratamento de simples aninhamentos sintáticos.
Linguagens que exibem características extensíveis podem beneficiar-se das propriedades dos autômatos adaptativos, os quais oferecem um caminho natural para a formalização sintática completa de características de extensibilidade, evitando a separação física usualmente observada na maioria das formalizações de linguagens extensíveis, entre as definições formais da linguagem hospedeira e as dos mecanismos de extensão.
Há diversos aspectos de extensibilidade para cuja representação formal os autômatos adaptativos se mostram extremamente aderentes do ponto de vista conceitual, o que torna conveniente o emprego de tais formalismos como pontos de partida para a obtenção, para tais linguagens, de implementações isentas dos incômodos pré-processadores encontrados, em geral, em muitas implementações tradicionais. Entre os mais importantes aspectos das linguagens extensíveis que se podem tratar através de técnicas adaptativas podem ser destacadas:

- declaração de macros simples e paramétricas
- declaração de macros isoladas, encadeadas e recursivas
- uso de parâmetros homônimos em macros paramétricas encadeadas
- respeito ao escopo dos nomes envolvidos nos tratamentos de macros
- chamada de macros no interior do corpo de macros
- chamada de macros no interior de argumentos de macros
- chamada, direta ou indiretamente recursiva, de macros

Ainda no que tange à extensibilidade lingüística, os autômatos adaptativos permitem superar algumas dificuldades usualmente encontradas na formalização de linguagens extensíveis, proporcionando ao seu usuário a possibilidade de efetuar:

- declaração e expansão de extensões funcionais
- declaração e expansão de macros sintáticas
- formalização completa da sintaxe de linguagens extensíveis

- definição semântica de extensões lingüísticas, em função de construções existentes

Os recursos conceituais de que o autômato adaptativo é dotado permitem ir além, utilizando-o como modelo computacional similar ao proporcionado pelas máquinas de Turing. Assim, com o emprego das transições do autômato torna-se possível estabelecer convenções para a criação de representações de conceitos muito básicos da teoria, tais como os seguintes:

- elemento nulo
- elementos constantes, escalares e vetoriais
- operação de obtenção do sucessor de um elemento
- operação de seleção do n-ésimo elemento de um vetor

Com isto torna-se possível proporcionar um modelo de computação baseado no paradigma funcional, através da composição adequada de tais operações, para a obtenção de outras mais poderosas.
Tal prática pode ser realizada com o auxílio das funções adaptativas e de suas chamadas paramétricas, com as quais é possível representar:

- composição de operadores
- substituições encadeadas de parâmetros por argumentos
- realização de cadeias de operações mutuamente recursivas

Com o auxílio das ações adaptativas básicas de inspeção torna-se possível ainda:

- comparar transições, e portanto as informações a elas associadas
- efetuar buscas de informações que estejam codificadas em um conjunto de transições

Pelo uso de ações básicas de eliminação e de inserção pode-se ainda introduzir o conceito de variável, abrindo caminho para a utilização do autômato adaptativo como um modelo operacional de computação, pois permitem alterar os elementos representados segundo as convenções adotadas, o que é possível, já que o emprego de tais ações básicas em funções adaptativas permite:

- eliminar uma ou mais transições
- inserir uma ou mais transições
- efetuar estas operações sobre partes do autômato escolhidas para representar variáveis

Podendo ser representado de forma natural como um grafo orientado, o autômato adaptativo permite mapear, em seu conjunto de transições, sem quaisquer artifícios, relacionamentos eventualmente existentes entre as informações representadas, permitindo o uso direto de sua própria estrutura para exprimir os tipos de dados usualmente empregados na programação imperativa, completando desta maneira o conjunto de recursos lingüísticos exigidos para a implementação de linguagens aderentes a esse estilo de programação.

7 Aplicações

É desnecessário comentar acerca da aplicabilidade do modelo proposto, dado que se trata de um autômato, com poder de máquina de Turing, e portanto muito adequado para uso como *modelo abrangente de computação*. No entanto, diversas aplicações particulares podem ser destacadas, por causa da naturalidade com que os autômatos adaptativos são capazes de representá-las e de tratá-las.

A primeira, obviamente, é a *representação de linguagens*, em particular as sensíveis ao contexto. Para esta aplicação, os autômatos adaptativos têm se mostrado bastante convenientes, e mais confortáveis que a maioria dos formalismos equivalentes disponíveis.
Como caso particular de grande relevância, destaca-se seu uso como *mecanismo de definição e de representação para linguagens naturais.*
A notação proposta para os autômatos adaptativos pode ser vista também como uma excelente *meta-linguagem textual*, interessante para denotar especificações de entrada em *ferramentas para a geração automática ou semi-automática de compiladores*.
Em adição, os autômatos adaptativos podem ser considerados não apenas modelos de computação capazes de representar a parte algorítmica dos programas, mas também, pela sua característica de máquina acionada por eventos, como fundamento conceitual adequado para a representação do *mecanismo básico de operação de interfaces homem-máquina*. Assim, trata-se realmente de um *substrato para a formalização de sistemas* em geral, cujos diversos casos particulares lhe permitem o uso, em certos casos, como forma alternativa para a *especificação formal de programas*.
Em particular, programas que podem ser modelados por meio de máquinas de estados, bem como programas dirigidos por eventos, são adequada e confortavelmente representáveis por meio de tal formalismo, que se mostra muito conveniente para a *representação de protocolos de comunicação*.
Ainda nestas aplicações à Engenharia de Software, os mecanismos adaptativos permitem que os autômatos adaptativos sejam utilizados como elementos de *suporte para a implantação de métricas*, e como substrato para a criação de *mecanismos de avaliação* do desempenho de sistemas de software.
Outra área que muito pode se beneficiar com os autômatos adaptativos é a Inteligência Artificial. Os autômatos adaptativos incorporam, na forma de transições adaptativas, um *mecanismo básico de aprendizagem*, com o qual tais máquinas se tornam capazes de memorizar informações acerca do histórico da sua operação, bem como de alterar seu próprio comportamento em função desse aprendizado.
Constitui desta maneira um *suporte para a construção de sistemas inteligentes*, com o auxílio do qual se torna possível elaborar *infra-estruturas para ambientes de ensino assistidos por computador*, suportes para *mecanismos de tomadas de decisão*, inferências, sistemas especialistas, aplicação em ambientes de *ensino semi-formal* assistidos por computador, e como substrato para a *representação de conhecimento e sua captura*.
Obviamente, a aplicação mais aderente ao caráter variável e adaptável do formalismo proposto é, evidentemente, voltada à *representação e realização de linguagens e sistemas inteligentes, dinâmicos, adaptáveis ou extensíveis*.
Neste tipo de emprego, os autômatos adaptativos podem ser vistos como um dos poucos formalismos completos que se encontram disponíveis para auxiliar na concepção, especificação e implementação confortável e natural de tal categoria de sistemas.
Não se tentou fazer aqui um levantamento exaustivo das inúmeras potenciais aplicações dos autômatos adaptativos, mas apenas um levantamento daquelas que se mostram mais significativas, do ponto de vista prático, para a resolução de problemas complexos, por procurarem explorar ao máximo as características do formalismo proposto que se mostram nitidamente diferenciadas em relação aos modelos convencionalmente utilizados na solução de tais problemas.
Cabe observar, como consideração geral sobre a proposta, a multidisciplinaridade potencial exibida pelo modelo, que, com suas bases fundamentadas na teoria do formalismo convencional subjacente, permite um tratamento natural para problemas de elevada complexidade intrínseca, como se pode observar em diversos dos casos anteriormente discutidos.

8 Contribuições

Para concluir, podem-se apontar as mais importantes contribuições desta pesquisa:

- uma notação unificada, aplicável a todos os tipos de autômatos
- um formalismo prático, com poder de representação de dependências de contexto
- potencial para a automatização da geração dos autômatos a partir de gramáticas
- um formalismo aderente às características das linguagens extensíveis
- um modelo intuitivo, adequado para ensino de teoria da computação em cursos não-teóricos
- um substrato conceitual para a obtenção de núcleos eficientes de compiladores
- potencial para um funcionamento muito eficiente do modelo
- fortes aplicações didáticas do modelo em cursos de engenharia
- diversas ferramentas que podem ser usadas como ambientes de desenvolvimento de implementações eficientes de compiladores

Diversas implementações-piloto efetuadas simularam a atuação do modelo em diversos casos:

- eliminação de pilha explícita em reconhecedores livres de contexto
- eliminação de tabelas explícitas de símbolos e de atributos
- eliminação de tabelas explícitas de palavras reservadas
- implementação exclusivamente sintática do tratamento de escopo em linguagens com visibilidade controlada de seus objetos, como é o caso das linguagens estruturadas em blocos
- implementação exclusivamente sintática de mecanismos de "type-checking" em linguagens fraca ou fortemente tipadas
- implementação sintática, sem pré-processamento, de mecanismos de extensão para linguagens extensíveis
- implementação sintática, de macros paramétricas encadeadas - definição e expansão

Informações complementares sobre a história e sobre os objetivos e metodologia adotados em diversos outros projetos e atividades relacionadas com a presente pesquisa estão descritas em [21].
Muitas outras contribuições poderiam ser listadas. Para obter uma visão muito mais completa e abrangente do alcance deste projeto, recomenda-se uma visita à home page do LTA – Laboratório de Linguagens e Tecnologias Adaptativas do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo – Brasil [11].

9 Referências

[1] Pagan, F.G. *Formal Specification of Programming Languages: A Panoramic Primer* Prentice-Hall, (1981).

[2] Slonneger, K. and Kurtz, B.L. *Formal Syntax and semantics of programming languages - a laboratory based approach* Addison Wesley, (1995).

[3] Shutt, J.N. *Recursive Adaptable Grammars* Ph D. Thesis, Worcester Polytechnic Institute, (1993)

[4] José Neto, J. *Contribuições para a Metodologia de Construção de Compiladores* - Tese de Livre Docência - Escola Politécnica da USP, (1990).

[5] José Neto, J. *Adaptive Automata for context-sensitive languages* ACM SIGPLAN Notices, v.29, n.9, p.115-24, Sept. (1994).

[6] José Neto, J. *Introdução à Compilação* Editora LTC, Rio de Janeiro, (1987).

[7] Almeida Jr., J.R. *STAD - Uma ferramenta para projeto e utilização de Statecharts Adaptativos* Tese de Doutoramento, Escola Politécnica da USP, (1995).

[8] JOSÉ NETO, J.; KOMATSU, W. *Compilador de Gramáticas Descritas em Notação de Wirth Modificada.* Anais EPUSP, Engenharia de Eletricidade, Série B, vol. 1, pp. 477-518, (1988).

[9] Taniwaki, C. Y. O. *Formalismos adaptativos na análise sintática de linguagem natural.* Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, (2001).

[10] Pereira, J. C. D. *Ambiente integrado de desenvolvimento de reconhecedores sintáticos, baseado em autômatos adaptativos.* Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, (1999).

[11] Home page do Laboratório de Linguagens e Tecnologias Adaptativas da Escola Politécnica da USP – S. Paulo – http://www.pcs.usp.br/~lta

[12] Pistori, H. and José Neto, J. *A Free Software for the Development of Adaptive Automata* IV Workshop sobre Software Livre – WSL 2003, Porto Alegre – Brasil (2003) pp.87-90.

[13] Harel, D. *Statecharts: a visual formalism for complex systems.* - Science of Computer Programming, v.8, n.3, p.231- 74, Aug. (1987).

[14] Santos, J. M. N. *Um formalismo adaptativo com mecanismo de sincronização para aplicações concorrentes*. Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, (1997).

[15] Iwai, M. K. *Um formalismo gramatical adaptativo para linguagens dependentes de contexto*. Tese de Doutorado, Escola Politécnica da USP, São Paulo, (2000).

[16] Basseto, B. A**.** *Um sistema de composição musical automatizada, baseado em gramáticas sensíveis ao contexto, implementado com formalismos adaptativos.* Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, (2000).

[17] Conway, M. E. *Design of a separable transition diagram compiler* Communications of the ACM,vol. 6, n. 7, Jul. (1963), pp 396-408.

[18] Lomet, D. B. *A formalization of transition diagram systems* Journal of the ACM, vol 20, n. 2, Feb. (1973), pp. 235-257.

[19] Rubinstein, R.S. and Shutt, J.N. *Self-modifying finite automata* Worcester Polytechnic Institute - Computer Science Department - Computer Science Technical Report Series - Technical Report WPI-CS-TR-93-11, december, (1993).

[20] Rubinstein, R.S. and Shutt, J.N. *Self-modifying finite automata - basic definitions and results* Worcester Polytechnic Institute - CS Department - Computer Science Technical Report Series - Technical Report WPI-CS-TR-95-2, august (1995).

[21] José Neto, J. *State of the Research on Adaptive Technology at the Escola Politécnica da Universidade de São Paulo* – WECI 2003 – I Workshop de Educación en Computación e Informática – Lima, Peru, agosto, (2003).