UM SUPORTE PARA A GERAÇÃO DE COMPILADORES

Mario Eduardo S. Magalhães
João José Neto

Escola Politécnica da Universidade de São Paulo
Caixa Postal 11455 - São Paulo - SP
Telefone: 211-2122 - Ramal 392

Palavras chave: Reconhecedores sintáticos, processadores de linguagens, gramáticas, compiladores, ferramentas para desenvolvimento e documentação de software, geração de compiladores com o auxílio do computador.

Resumo:

Relata-se a operação e organização de um conjunto de programas desenvolvidos no LSD da EPUSP com a intenção de gerar suporte para desenvolvimento de processadores de linguagens de programação. Esta infraestrutura consta globalmente de um conjunto de programas encarregados de aceitar como entrada, no caso geral, uma gramática livre de contexto da linguagem cujo processador se deseja construir, produzindo um reconhecedor para tal linguagem, que opera como núcleo do processador desejado. Este reconhecedor pode ser produzido por duas técnicas diferentes, à escolha do usuário, podendo opcionalmente operar em cooperação com pré-processadores de linguagem configuráveis, também fornecidos pela infraestrutura.
O conjunto de programas acima citado foi criado como atividade de suporte para o desenvolvimento do projeto SPD (9), tendo sido intensamente utilizado durante a fase de implantação do mesmo. Atualmente está sendo realizada a integração destes programas com a finalidade de compor um subsistema do próprio SPD para auxílio à geração de compiladores.

1. Introdução

O presente texto tem por objetivo expor funcional e estruturalmente os programas que compõem uma infraestrutura para o desenvolvimento de processadores de linguagens, explorando as características e funções do conjunto, quer como parte de um sistema de construção de compiladores a ser implantado, quer isoladamente, ou ainda na forma de combinações funcionais. Inicialmente é feito um pequeno histórico, onde é relatada a origem e a motivação do projeto, implementação de tais programas, e ambiente lógico em que operam. A seguir são estudados os objetivos globais tecendo-se considerações acerca de algumas possíveis combinações funcionais. É então apresentada, para cada módulo, sua descrição estrutural e funcinal, finalizando-se em considerações acerca das características do programa construído com auxílio desta infraestrutura.

2. Histórico

Em meados de 1980 surgia a primeira versão operacional do SPD (9) Sistema Automático de Apoio ao Projeto e Desenvolvimento de Sistemas Digitais. Nesta ocasião contava o SPD com recursos implantados para a construção automática de simuladores e programas congêneres, e existia, já projetado, em início de implantação, uma ampliação destes recursos visando a geração automática de cross-montadores para as máquinas simuladas.
Para a construção dos processadores de linguagem utilizados neste protótipo (que envolve duas linguagens de alto nível e dois métodos de processamento: o compilador LD e o interpretador LE) foram utilizados duas técnicas (respectivamente: produções de Floyd-Evans (6,7) e tabelas de transição (8)).
O desenvolvimento destes processadores, inicialmente manual, exigiu, para a obtenção de resultados em tempo hábil, a elaboração de vários programas auxiliares que viabilizaram a construção confiável de partes do sistema.
Expansões subseqüentes da capacidade de SPD vieram a confirmar a utilidade de tais programas e a motivar a exploração dos mesmos, não mais como módulos isolados, mas como integrantes de um pequeno sistema, que incorpora novos programas visando complementar seus recursos e dotá-los de maior versatilidade.
Atualmente o conjunto de programas está sendo integrado e deverá ser incorporado ao SPD como parte de um sistema semi-automático de construção de compiladores.

3. Objetivos

Pode-se agrupar os componentes desta infraestrutura nos seguintes conjuntos de programas, do ponto de vista funcional:
a) **Transdutores**: efetuam as conversações entre a gramática, descrita em BNF, e um formato interno, entre tal formato e outros que representam o reconhecedor desejado conforme especificação do usuário, ou que ainda convertem tais representações para a linguagem ALGOL do sistema Burroughs B6700 (3), gerando programas executáveis que implementam o reconhecedor. Tal conjunto de programas tradutores constituem o suporte desta infraestrutura e, sob comando do usuário, podem acionar outros módulos, visando maior agilidade na construção do processador de linguagem desejado.
b) **Processadores da gramática:** responsabilizam-se pela análise e documentação da gramática de entrada emitindo listagens e avisos que facilitam posteriores referências e modificação na gramática.
c) **Interpretadores:** encarregados de utilizar diretamente as representações internas do reconhecedor desejado afim de tornar possível o reconhecimento sintático da linguagem descrita.
d) **Pré-processadores:** fornecem, para os reconhecedores construídos automaticamente, rotinas de apoio, tais como macro-expansores e processadores de comandos de controle.
A operação desta infraestrutura pode ser efetuada partindo-se de uma gramática livre de contexto. Esta, quando fornecida ao sistema, em uma primeira etapa, é analisada quanto à consistência, conexidade e outras características, obtendo-se listagens que visam o seu desenvolvimento e documentação.
Uma segunda etapa de processamento permite ao usuário a possibilidade de ensaiar o comportamento da gramática fornecida em relação às dificuldades de construção do processador da linguagem dela derivada. O ensaio da gramática é efetuado pelo usuário, alternadamente com modificações da mesma, visando a obtenção, para a linguagem, de uma gramática concisa e que simultaneamente propicie a geração automática de um reconhecdor eficiente e de fácil utilização. Obtida a gramática final, o sistema é capaz de gerar automaticamente um reconhecedor sitático que constituirá a base do processador de linguagem desejado.
Com tal finalidade, o sistema dispõe de dois modos de operação: o primeiro utiliza a geração de reconhecedores "top-down", baseados em tabelas de trasição construídas automaticamete pelo sistema e que implementam um reconhecedor (usualmente LL (1)) da linguagem em questão, sempre que possível gerando indicações das estruturas cujo mecanismo de reconhecimento não pode ser construído por este método. Esta implementação constitui um caso particular de um gerador de reconhecedores sintáticos para gramáticas livres de contexto gerais (1,2,8,10,11), que se encontra em desenvolvimento e constituirá a versão definitiva a ser implantada no SPD.
O segundo método utiliza uma adaptação do algoritmo proposto por Earley (5) para geração de uma versão intermediária do reconhecedor desejado expresso na forma de uma seqüência de produções de Floyd-Evans modificadas.
Em ambos os casos a geração automática produz versões do reconhecedor em formato fonte, o que implica na necessidade de um processamento prévio do mesmo a fim de tornar econômico o reconhecimento. Durante a fase de desenvolvimento da gramática não se exige, obviamente, que o reconhecedor tenha operação eficiente, sendo recomendável, nesta fase, a utilização de métodos interpretativos das formas intermediárias do reconhecedor (produções de Floyd-Evans ou tabelas de transição). Atingido para a gramática um estado de desenvolvimento julgado satisfatório, o usuário tem a possibilidade de, através da aplicação de programas tradutores específicos fornecidos pela infraestrutura, construir versões diretamente compiláveis pelo sistema hospedeiro (no caso, em ALGOL do B6700). Este processo de tradução envolve, principalmente para as produções de Floyd-Evans modificadas, uma etapa prévia de refinamento do reconhecedor final construído, visando maior velocidade de execução e portanto menor custo operacional.
A incorporação de rotinas semânticas ao reconhecedor pode ser feita durante a fase iterativa do processo descrito, ocasião em que o usuário tem condições de informar ao sistema as construções sintáticas, semanticamente ativas. Desde modo, o sistema tem a possibilidade de gerar processadores da linguagem definida, que possuam maior eficiência para os casos informados pelo usuário.
Como recursos adicionais, o sistema oferece ainda alguns módulos que, a critério do usuário, podem ser configurados para o caso particular que se deseja

explorar e incorporados ao processador básico obtido, isolada ou conjuntamente, facilitando assim a obtenção de processadores de linguagem com recursos poderosos de pré-processamento. Nesta categoria pode-se citar um analisador léxico geral, capaz de reconhecer construções definíveis através de expressões regulares; um macro-expansor de uso geral, responsável pela pré expansão de macros definidas pelo usuário do compilador a ser gerado; um processador de comandos de controle, responsável pela realização do controle de opções de compilação, tais como: listagens, geração de tabelas de referências cruzadas, inclusão de arquivos, etc.

4. Detalhamento

Descreve-se a seguir os módulos de que se compõe o sistema do ponto de vista de funcionamento e de estrutura.
Como foi mencionado, o sistema se compõe basicamente de módulos encarregados do sucessivo mapeamento entre representações de uma linguagens, partindo-se de uma gramática inicial até a obteção final de um processador de linguagem desejado. Desta maneira, a descrição do sistema será efetuada, por razões didáticas, apresentando-se um a um os vários módulos tradutores, a conversão de formatos por eles implementada, e os módulos restantes que a eles são relacionados.
A figura 1 mostra, esquematicamente, as inter-relações mais freqüentes na utilização dos módulos do sistema.

4.1. Tradução de BNF para Formato Interno

Apesar de ser possível ao usuário acessar os recursos do sistema a partir de qualquer dos formatos intermediários, é mais conveniente limitar este acesso ao fornecimento de uma descrição formal da sintaxe da linguagem desejada em BNF. Esta gramática sofre então sua primeira tradução, obtendo-se um formato interno que mapeia biunivocamente as informações contidas no texto BNF para uma representação de mais conveniente manipulação pelas etapas subseqüentes de processamento a que poderá ser submetida.
Esta tradução é efetuada em um único passo, onde os meta-identificadores e os terminais da linguagem descrita são coletados em uma tabela, associando-se a cada qual uma identificação numérica única.
As inter-relações definidas no texto BNF são transformadas em cadeias de apontadores, de acordo com o mostrado na figura 2.
Nesta etapa do processamento, deve-se observar que o texto BNF sofre uma abreviação com a finalidade de agrupar produções que definem os mesmos meta-identificadores.
A dupla ocorrência de uma mesma produção, bem como a inexistência de produções que definam um determinado meta-identificador são condições anormais, reportadas ao usuário pelo sistema.
A lógica que implementa este mapeamento é, como se pode facilmente intuir, extremamente simples, e não será descrita neste texto.
O usuário pode ativar nesta fase dois módulos satélites, a saber: um gerador de referências cruzadas, e um módulo encarregado de testar a consistência da gramática fornecida, recebendo destes procedimentos mensagens de aviso acerca de possíveis inconsistências nela detetadas. O usuário pode então interagir com o sistema, efetuando alterações na gramática original com a finalidade de eliminar as inadequações detetadas. A representação em formato interno pode então ser considerada pronta para ser submetida aos demais módulos tradutores disponíveis no sistema.

4.1.1. Gerador de Referências Cruzadas

Ao ser realizada a leitura da gramática o sistema numera univocamente as produções BNF fornecidas emitindo uma listagem para documentação.
Adicionalmente o usuário tem a possibilidade de solicitar ao sistema a execução do módulo gerador de referências cruzadas que completa tal documentação com a construção e emissão de um índice remissivo de todos os símbolos utilizados na gramática: Neste índice é feita uma distinção entre as produções em que os meta-identificadores são definidos e aquelas onde são referenciados. Observe-se que o par de listagens assim emitido constitui um recurso extremamente valioso para auxílio ao usuário na tarefa de depuração da gramática facilitando a localização e a correção de possíveis inadequações da gramática inicial, bem como posteriores alterações da mesma quando conveniente para o usuário.
O algoritmo de construção da tabela de referências cruzadas utilizado no sistema pode ser considerado clássico, sendo que as estruturas de dados estendem a tabela de símbolos apresentada na figura 2 através da incorporação de listas ligadas responsáveis pela implementação dos conjuntos de referências às diversas ocorrências dos símbolos no texto BNF. A figura 3 esboça esta organização dos dados.

4.1.2. Testes de Consistência

Para ser possível aos módulos subseqüentes a execução dos algoritmos de geração do reconhecedor a partir da gramática é indispensável que o texto BNF fornecido representa uma gramática reduzida, isto é, que não contenha definições cíclicas e que seja conexa. Assim sendo, o sistema fornece ao usuário um diagnóstico da ocorrência de tais impropriedades da linguagem, solicitando a eliminação destas para prosseguimento da operação.
Com a finalidade de validar a gramática fornecida este módulo opera sobre o produto da execução do programa tradutor BNF para formato interno (figura 2).
A análise de redução da gramática é efetuada através de um procedimento recursivo que verifica a existência de seqüências de substituições que levam da raiz da gramática a todos os elementos (terminais e não terminais) utilizados no texto BNF fornecido.

4.2. Tradução do Formato Interno para Produções de Floyd-Evans

A primeira metodologia disponível no sistema para a obtenção de um reconhecedor a partir da gramática fornecida consiste em mapear automaticamente o texto BNF, agora em formato interno, para um conjunto de produções de Floyd-Evans modificadas baseando-se para isto no método clássico apresentado detalhadamente por Earley em seu trabalho sobre reconhecedores sintáticos (5). O algoritmo básico foi adaptado às necessidades do sistema especialmente no que tange à notação e formatos utilizados.
Como produto este módulo coloca a disposição do usuário a descrição de um reconhecedor sintático em produções de Floyd-Evans modificadas. Estas produções são impressas para efeito de documentação e desenvolvimento, caso haja interesse do usuário, e uma cópia das mesmas permanece disponível no sistema para futuras manipulações (automáticas) se necessário. Note-se que neste ponto o sistema já transformou a gramática fornecida em um reconhecedor cuja descrição pode ser utilizada diretamente para a construção automática do programa reconhecedor sintático desejado. Para tanto dois métodos encontram-se implementados no primeiro deles a seqüência de produções é encarada como um roteiro para o reconhecimento, uma vez que fornecem indicações das ações a serem tomadas durante o reconhecimento em cada caso que possa ocorrer no texto fonte. No segundo método as produções são convertids em um programa executável que efetua as ações de reconhecimento necessárias através de uma linguagem de alto nível (ALGOL do sistema B6700). Ambos os módulos serão descritos adiante.

4.2.1. Módulo de Consistência

Cumpre notar que é possível fornecer produções de Floyd-Evans modificadas construídas externamente, como método alternativo de entrada, permitindo assim alimentar o sistema diretamente com um reconhecedor ao invés de com uma gramática. Neste caso o formato das produções de Floyd-Evans se comporta como uma linguagem de entrada do sistema. Assim, como suporte a este recurso dispõe-se de um módulo que efetua a consistência das produções manualmente introduzidas, encarregando-se de testar sua validade sintática e fornecer mensagens de erro ao usuário caso alguma inconsistência seja detetada.
Há dois tipos de consistência testados nas produções por este módulo: no primeiro, o formato de cada produção individual é verificado, observando-se a sintaxe esperada para estas produções. São fornecidas mensagens de erros específicos caso tal sintaxe não seja obedecida. No segundo teste, uma análise mais global é efetuada na seqüência de produções, verificando-se dupla definição ou identificação de rótulos produções inatingíveis pela ordem em que aparecem, produções inatingíveis por não serem referenciadas, e outros erros de contexto semelhantes. São neste caso fornecidas mensagens de erro, posteriores à listagem das produções.

4.3. Tradução de Formato Interno para Tabelas de Transição

Este módulo tradutor implementa o segundo método de obtenção automática de reconhecedores sintáticos oferecido pelo sistema. Apresenta-se, atualmente, em duas versões: na primeira,é realizada a construção de reconhecedores para linguagens restritas, e na segunda, para linguagens livres de contexto gerais. No caso restrito, as produções são analisadas e a partir desta análise são construídas diretamente tabelas de transição que fornecem um roteiro para o reconhecimento top-down da linguagem gerada pela gramática BNF fornecida. Observe-se que deste modo são impostas retrições quanto à gramática que pode ser processada, restrições estas decorrentes da maneira pela qual são obtidas as tabelas de transição: o conjunto de produções é analisado em um único passo, sendo esta análise efetuada elemento a elemento na cadeia dos meta-símbolos que compõem a gramática fornecida (agora em formato interno). Por simplicidade de construção, da análise efetuada deriva-se um conjunto de tabelas de transição cuja construção é incremental, sendo que novas tabelas, novas colunas, novas linhas e o preenchimento de células vazias da parte já construída das tabelas ocorrem somente na ocasião da análise de cada um destes elementos, o que acarreta uma limitação na classe de gramáticas tratáveis pelo método: gramáticas LL (1) ou então as que não o são exclusivamente devido à

presença de produções BNF reursivas à esquerda. Esta não é uma restrição forte à utilização do sistema, já que a maioria das linguagens de programação de interesse pode ser descrita através de gramáticas que obedeçam a esta restrição.
Como saídas, esta versão do programa oferece a listagem impressa das tabelas construídas, avisos de ocorrência de construções não tratáveis pelos algoritmos do sistema, e uma versão interna em disco das tabelas construídas.
O segundo método para a geração de tabelas de transição suprime a restrição mencionada anteriormente, uma vez que é suficientemente poderoso para aceitar e tratar qualquer gramática livre de contexto. Este método, efetuado em vários passos, envolve uma série de transformações sobre a gramática fornecida, e é fundamentado na teoria descrita em (10) e (11).
Observe-se que os dois métodos oferecidos pelo sistema produzem saídas em formatos compatíveis. As tabelas de transição produzidas não são propriamente o reconhecedor desejado, embora sejam descrições formais de seu funcionamento. Portanto, exigem, para implementar o reconhecedor, um módulo tradutor das tabelas para formato executável, ou então um programa interpretador destinado à execução das transições adequadas.

4.3.1. Interpretador de Tabelas de Transições

A interpretação das tabelas de transição construídas automaticamente torna possível a implementação de um reconhecedor bastante simples que dispensa novas traduções das tabelas de transições, sendo portanto indicado durante a fase de depuração e desenvolvimento da gramática.
A operação do interpretador é extremamente simples constando basicamente em efetuar transições internas a uma tabela, chamadas de uma tabela como "sub-rotina" de outro (tabela corrente) ou então retorno para a tabela anterior. A decisão de qual operação deve ser realizada é efetuada pelo programa com base no conteúdo de uma célula da tabela, associada a um estado corrente e a um elemento léxico do texto em que se efetua o reconhecimento sintático, conforme esquematizado na figura 4.
A cada transição pode estar associada a execução de um procedimento fornecido pelo usuário (ação semântica), a especificação dessas ações semânticas pode ser realizada quando da análise da gramática ou diretamente sobre as tabelas construídas.

4.4. Tradução de Produções de Floyd-Evans para Formato Interno Interpretável

Dado um conjunto de produções de Floyd-Evans que constitui a descrição formal do reconhecedor sintático, sendo portanto um roteiro das ações de reconhecimento, é possível no sistema aqui apresentado efetuar uma transformação notacional através de um módulo tradutor, que tem por finalidade criar um conjunto equivalente de produções em formato manipulável eficientemente por um intepretador dirigido por tal conjunto de regras.
Este tradutor é estruturalmente semelhante a um montador convencioal ("assembler"), onde o programa fonte se constitui das produções em formato externo, e o programa objeto é um conjunto de vetores que contêm, em formato compacto e de acesso direto, o conjunto de informações de interesse contido na forma externa das produções de Floyd-Evans. No formato interno, são eliminados os rótulos simbólicos, sendo estes, onde necessário, substituídos por referências numéricas.

4.4.1. Interpretador de Produções de Floyd-Evans

Na construção de um reconhecedor sintático pelo método interpretativo, o módulo interpretador de produções de Floyd-Evans constitui o último estágio de processamento, que implementa finalmente o reconhecedor desejado. Este módulo é alimentado pelas produções traduzidas para formato interno, obtidas anteriormente, e que funcionam como uma tabela de decisões para o programa interpretador.
Analogamente ao interpretador de tabelas de transição, a decisão da ação de reconhecimento a ser efetuada é baseada em um elemento léxico extraído do texto que se deseja reconhecer, e em uma seqüência conveniente de opções de reconhecimento, cada qual constituída de uma produção de Floyd-Evans, e que fornece todas as possibilidades de reconhecimento permitidas em uma dada situação. Escolhida, através de uma produção, a ação de reconhecimento a ser adotada, uma ação semântica a ela associada é executada caso esteja presente uma indicação neste sentido.
A especificação das ações semânticas que devem ser executadas no reconhecimento sintático é efetuada pelo usuário quando da análise da gramática, ou diretamente sobre as produções de Floyd-Evans.

4.5. Tradutor de Produções de Floyd-Evans para ALGOL

A utilização do primeiro método (interpretativo) descrito anteriormente é ineficiente devido ao grande número de buscas necessárias nas estruturas de dados, estas de tamanho considerável, e que correspondem às produções de Floyd-Evans. Para a obtenção de reconhecedores definitivos, há uma grande conveniência de eliminar quando possível tais deficiências, o que é implementado pelo segundo método, que efetua a tradução das produções de Floyd-Evans para uma linguagem de alto nível (ALGOL do B6700).
O tradutor em questão pode ser considerado como um macro-expansor particular que, quando executado, gera, a partir das produções (consideradas como chamadas de macros), textos expandidos em linguagem ALGOL do B6700 conforme mostra graficamente a figura 4. A lógica de expansão, como pode ser notado, consta de substituição textual, sendo portanto trivial.
O número de comparações necessário a um reconhecimento feito através do programa assim gerado é comparável ao de buscas nas estruturas de dados utilizada no método interpretativo. Como as buscas no método interpreativo são efetuadas em tempo de execução, e no método ora apresentado, são explicitadas através da geração de comparações encadeadas, conclui-se que o tempo de execução neste último caso terá grande probabilidade de ser mais eficiente. Um mapeamento intermediário das produções em árvores binárias pemite reduzir ainda mais o número de comparações geradas, melhorando a eficiência global do programa gerado.
Com esta fase de redução de comparações, o programa tradutor passa a operar em dois passos:
- no primeiro, são separadas as seções qaue compõem o texto de produções de Floyd-Evans, e montadas matrizes que as representam.

- no segundo, tem-se duas fases, que são repetidas para cada seção:
a) construção, para cada seção, de uma árvore binária equivalente.
b) tradução da árvore para comandos ALGOL convenientes.
Neste segundo passo também são geradas todas construções ALGOL necessárias para completar simbolicamente o texto gerado.
A figura 5 exemplifica a geração de código ALGOL a partir de uma seção, com a construção intermediária da árvore binária correspondente.
O programa ALGOL assim construído constitui um reconhecedor sintático de bom desempenho correspondente ao conjunto de produções fornecidas inicialmente.
Este programa foi extensivamente utilizado no projeto SPD 191.

4.6 Tradutor de Tabelas de Transição para ALGOL

Sendo interpretativo o método de tabelas de transição anteriormente descrito, sua operação pode ser melhorada em eficiência introduzindo também neste caso uma tradução similar à que foi mencionada para as produções de Floyd-Evans.
Esta tradução consiste basicamente em associar aos estados da tabela rótulos em um programa ALGOL que deseja gerar, sob cada rótulo é construída uma sequência de decisões que de acordo com a entrada corrente desvia para o rótulo associado ao estado destino. A figura 6 representa o mapeamento acima descrito para um pequeno exemplo. Um pequeno conjunto de operações primitivas completa a lógica do programa.
Observa-se que tal tradução pode ser efetuada sobre as tabelas de transição geradas por qualquer um dos dois métodos disponíveis no sistema.
Este programa encontra-se em desenvolvimento devendo ser em futuro próximo incorporado ao sistema.

4.7 Pré-Processadores e Programas Auxiliares

Como auxílio ao desenvolvimento de processadores completos de linguagens (interpretadores, compiladores), dispõe-se, como ferramenta de apoio, de módulos de grande aplicabilidade visto serem diretamente acopláveis aos reconhecedores sintáticos automaticamente gerados, quando solicitado pelo usuário. Nesta classe de módulos dispõe-se basicamente de um macro-expansor e de um interpretador de comandos em tempo de compilação, um gerador de referências cruzadas, além de um gerador de analisadores léxicos e da possibilidade de inserção de rotinas semânticas cuja execução deverá ser controlada pelo analisador sintático obtido automaticamente.

4.7.1 Interpretador de Comandos em Tempo de Compilação

Através desse módulo é dada ao usuário a possibilidade de utilizar-se de recursos que lhe permitam controlar algumas das atividades do processador de linguagem, entre os quais pode-se citar: salto de páginas, salto de linhas, supressão opcional da emissão da listagem do programa fonte, controle da produção e emissão de tabelas de referências cruzadas, seleção da origem do texto a ser analisado, controle de compilação condicional, seleção do macro-expansor, etc.
Este programa não necessita de bases de dados, pois sua função consiste basicamente em analisar comandos de controle, identificá-los, interpretando-os ou executando ações a eles associadas ou ativando outros módulos auxiliares do sistema.
Tal módulo encontra-se atualmente em desenvolvimento constituindo um dos principais programas responsáveis pela integração do sistema.

4.7.2 Macro-expansor

Este módulo foi projetado para permitir ao usuário do processador de linguagem a ser construído, a definição e a utilização de macros textuais [4]. Consiste de

um processador de textos de uso geral com um pequeno número de comandos de controle, que efetuam inicialmente a descrição dos caracteres de controle a serem utilizados pelo expansor na definição e referências às macros utilizadas pelo usuário.

As definições de macros criadas pelo usuário são armazenadas em um arquivo de definições, apontado por uma tabela de símbolos. No arquivo de definições são armazenadas, em uma área de definições permanentes, os corpos dos macros definidos inicialmente, como também o texto a ser expandido.

Na tabela de símbolos são armazenados os nomes das macros juntamente com apontadores para a área de definições permanentes; os nomes dos parâmetros formais juntamente com um apontador para área de definições temporárias onde comparecerá o texto pelo qual o parâmetro deverá ser susbtituído durante a realização da expansão de macro correspondente. Note-se que a área de definições temporárias é dinâmica, estruturada em pilha permitindo assim chamadas encadeadas de macro, inclusive no texto de um parâmetro. Na versão atual não é permitida a chamada recursiva de macros.

O programa expansor consta de apenas um procedimento recursivo que expande uma macro através da cópia do corpo de sua definição para um arquivo de saída, realizando sua ativação recursivamente sempre que for detetado, durante a expansão, uma chamada de macro. Observa-se que o procedimento é iniciado pela expansão do texto principal, considerado como uma macro sem nome, e que em toda chamada paramétrica são criadas definições temporárias para os seus parâmetros os quais por sua vez são denotados e tratados, no corpo da macro, como chamadas de outros macros.

Um esboço das estruturas de dados utilizadas é mostrado na figura 7.

4.7.3 Gerador de Referências Cruzadas

A potência de um processador de linguagem é aumentada substancialmente se este dispuser de um gerador de referências similar ao descrito em 4.1.1.. Com essa finalidade foi desenvolvido para operar em conjunto com processadores de linguagem, particularmente aqueles obtidos com o auxílio desta infraestrutura, um gerador de referências cruzadas de propósito geral que opera cooperativamente com o analisador léxico utilizado pelo processador em quetão, quando solicitado pelo usuário através de um comando de compilação adequado. Estruturalmente o procedimento consta de duas seções, uma delas responsável pela criação e manutenção da base de dados conforme descrito anteriormente e a segunda que realiza a impressão da tabela construída.

4.7.4. Gerador de Analisadores Léxicos

Sendo essencial a qualquer processador de linguagem e disponibilidade de um analisador léxico compatível, torna-se interessante oferecer ao usuário desta infraestrutura uma ferramenta que o auxilie a desenvolver seu próprio analisador léxico, compatível com os demais módulos da infraestrutura.

Tal gerador é operado fornecendo-se uma gramática que define através de expressões regulares [1,2]na forma dos itens léxicos da linguagem em questão. A lógica do programa gerador consiste em criar uma implementação recursiva descendente [2,8] de um reconhecedor sintático diretamente a partir de expressões regulares, produzindo assim em linguagem ALGOL do B6700 um programa correspondente ao atômato finito descritos pelas expressões regulares fornecidas.

4.8. Considerações Acerca da Inserção de Ações Semânticas

Os módulos anteriormente descritos fornecem ferramentas para a obtenção automática de reconhecedores sintáticos completos para uma linguagem definida pelo usuário. No entanto, este produto é insuficiente quando se deseja obter processadores operantes para estas linguagens, uma vez que aqueles não apenas reconhecem construções sintáticas da linguagem, como a estas associam significados, representados por meio de execução de ações semânticas, usualmente efetuadas como decorrência do reconhecimento de tais construções em processadores dirigidos por sintaxe, como é o caso dos produzidos por intermédio desta infraestrutura.

Esta deficiência é atenuada mediante a exploração de alguns recursos fornecidos pelo sistema. Como foi mencionado, durante a tradução dos textos BNF para formato interno, o usuário tem a possibilidade de efetuar alterações na gramática fornecida visando a obtenção das estruturas sintáticas mais convenientes à inclusão de referências a procedimentos semânticos fornecidos pelo usuário, os quais podem ser incluídos durante a análise das produções de Floyd-Evans ou das tabelas de transição construídas pelo sistema, conforme o método escolhido.

Quanto às rotinas que implementam as ações semânticas, devem ser fornecidas pelo usuário em um formato compatível com o sistema, a saber, na forma de um procedimento com um único parâmetro que identifica a particular ação semântica a ser executada.

5. Considerações Finais

O presente texto limitou-se a apresentar informal e qualitativamente uma infraestrutura de apoio à construção de processadores de linguagens. Tal infraestrutura, em sua concepção final, é um sistema interativo integrado, voltado ao desenvolvimento de processadores de linguagens de alto nível em todas as suas etapas. Atualmente, vários de seus módulos encontram-se em funcionamento como programas independentes e outros em estágio de implementação. Simultaneamente, está em curso a integração dos diversos módulos, visando, em etapa posterior, sua inclusão no SPD, na forma de uma extensão responsável pelo apoio à construção semi-automática de compiladores.

Bibliografia

1. Barret, W.A. and J.D. Couch
"Compiler Construction - Theory and Practice" - SRA, 1979
2. Bauer, F.L. and J. Eickel (ed.)
"Compiler Construction - An Advanced Course" (2nd-Edition) Springer Verlag, 1976
3. Burroughs Corporation
"B7000 - B6000 Extended Algol Language Reference Manual"
4. Cole, A.J.
"Macro Processors"
Cambridge University Press, 1976
5. Early, J.
"An Efficient Context - free parsing algorithm"
Communications of the ACM vol. 13 nº 2 FEB/1970
6. Evans, Jr. A.
"An Algol 60 Compiler"
Annual Review in automatic Programming vol. 4 (1964)
7. Floyd, R.W.
"A descriptive Language for Symbol Manipulation"
Journal of the ACM vol. 8 nº 4 october 1961
8. Gries,D.
"Compiler Construction for Digital Computers"
John Wiley e Sons, Inc., 1971
9. Neto, J.J.
SPD - Um Sistema Automático de Apoio ao Projeto e Desenvolvimento de Sistemas Digitais"
Tese de Doutoramento - EPUSP - 1980
10. Neto, J.J. e Magalhães, M.E.S.
"Um Gerador Automático de Reconhecedores Sintáticos para o SPD" VII SEMISH, Florianópolis, 1981
11. Neto, J.J. e Magalhães, M.E.S.
"Reconhecedores Sintáticos - Uma alternativa didática para uso em Cursos de Engenharia"
XIV Congresso Nacionalde Informática - São Paulo, 1981

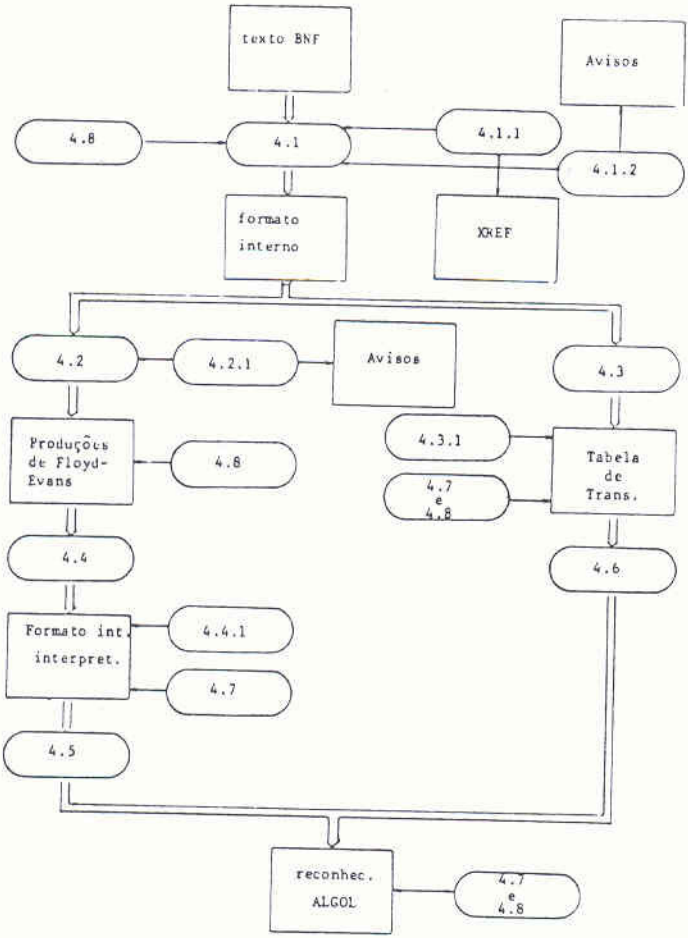


Figura 1.ª - Diagrama de Relações entre os Módulos Componentes da Interface (ver legenda)

Legenda

4.1. Tradutor BNF para Formato Interno
4.1.1. Gerador de Referências Cruzadas
4.1.2. Testes de Consistência
4.2. Tradutor de Formato Interno para Produções de Floyd-Evans
4.2.1. Módulo de Consistência
4.3. Tradutor de Formato Interno paar Tabelas de Transição
4.3.1. Interpretador de Tabelas de Transição
4.4. Tradutor de Produções de Floyd-Evans para Formato Interno Interpretável
4.4.1. Interpretador de Produções de Floyd-Evans
4.5. Tradução de Produções de Floyd-Evans para ALGOL
4.6. Tradução de Tabelas de Transição para ALGOL
4.7. Pré-Processadores e Programas Auxiliares
4.8. Inserção de Semântica

Fig. 1b - Diagrama de Relações entre os Módulos Componentes da Infraestrutura

Texto BNF:

<S>::= <A> | <B><C> | X <S> Y

<A>::= <A> X |Y| <C>

<B>::= <A> | Y <B>

<C>::= X | ε

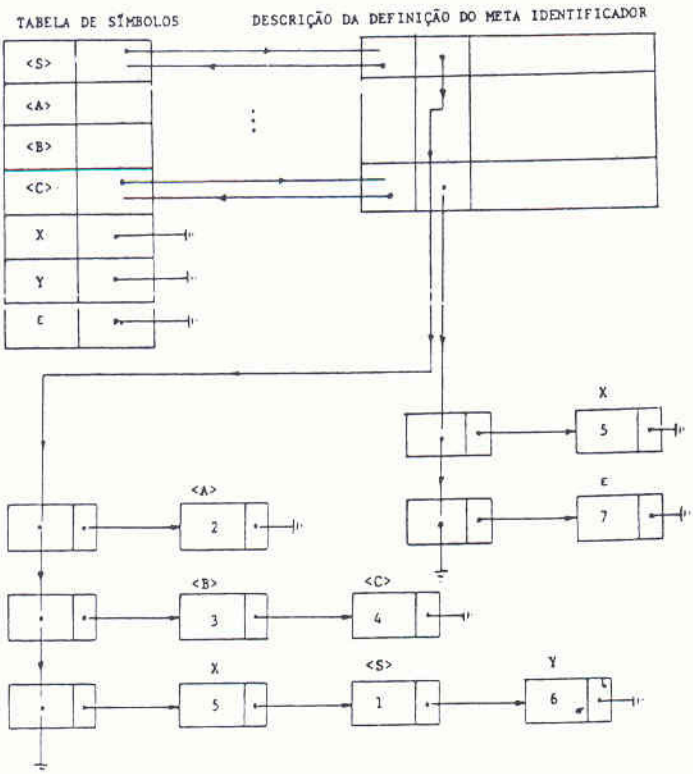
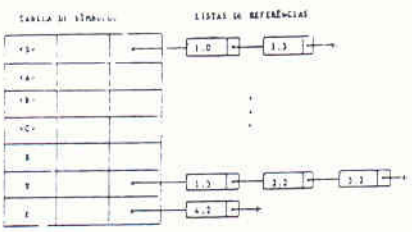


Figura 2 - Representação Esquemática do Formato Interno do Texto BNF em um Exemplo.



```
1.0     <S>   ::=
1.1               <A>
1.2             | <B> <C>
1.3             | X  <S> Y
2.0     <A>   ::=
2.1               <A>  X
2.2             | Y
2.3             | <C>
3.0     <B>   ::=
3.1               <A>
3.2             | Y  <B>
4.0     <C>   ::=
4.1               X
4.2             | ε
```

Listagem BNF formatada

Figura 3 - Estruturas de Dados do Gerador de Referência Cruzadas

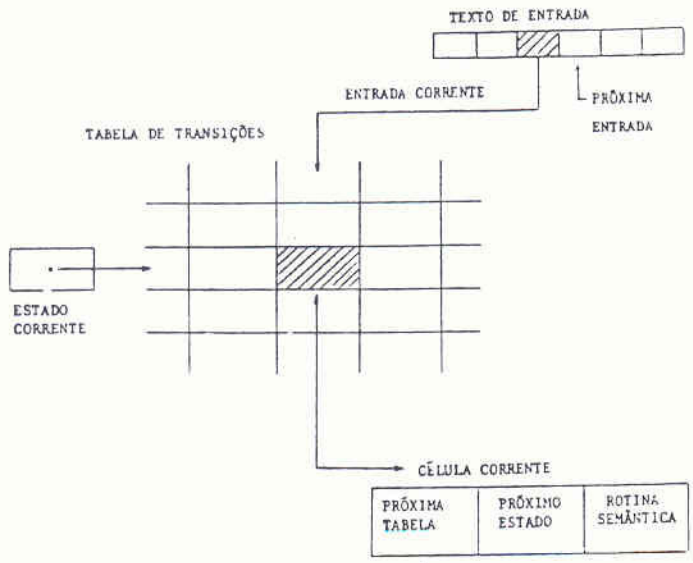


Figura 4 - Esquema do Funcionamento das Tabelas de Transições

OPÇÕES:

- a transição é interna (nenhuma especificação neste campo)
- ativação de uma tabela cuja identificação é especificada neste campo
- final de operação da tabela corrente e retorno à tabela anterior
- final de operação sem sucesso da tabela corrente.

SEÇÃO:							
	d	b	a	=	d	A	*A1
	f	e	a	=	A	a	A1
			c	=	A		*A1
			d				BØ
			T				Erro

Árvore binária:

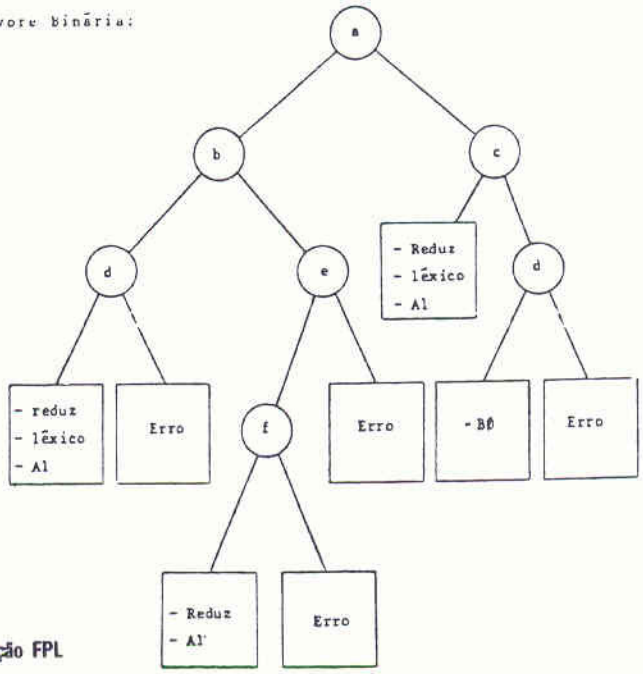


Figura 5a. - Geração de Código ALGOL a Partir de uma Seção FPL

Figura 5b. - Geração de Código ALGOL a Partir de uma Seção FPL

```
SEÇÃO: if (entrada) = "a"
       then  if (entrada-1) = "b"
             then  if (entrada-2) = "d"
                   then Begin
                        reduz ("d", "A");
                        léxico;
                        go to A1;
                        End
                   else Erro
             else  if (entrada-1) = "e"
                   then  if (entrada-2) = "f"
                         then Begin
                              reduz ("A", "a")
                              go to A1;
                              End
                         else Erro
                   else Erro

else  if (entrada) = "c"
      then Begin
           reduz ("A");
           léxico;
           go to A1;
           End
      else if (entrada) = "d"
           then go to BØ;
           else Erro:
```

Figura 6 - Tradução de uma Tabela de Transições para Linguagem Semelhante ao ALGOL Utilizando como Exemplo um Analisador Léxico Simplificado

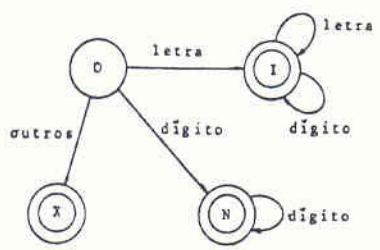


Diagrama de Estados

	letra	dígito	outros
0	I	N	X
I	I	I	retorna
N	retorna	N	retorna
X	retorna	retorna	retorna

Tabela de Transições

```
D:  if  entrada é letra
      then Begin
           avança;
           go to I;
           end
      else if entrada é dígito
           then Begin
                avança;
                go to N;
                end;

    avança;
    go to X;

I:  While entrada é (dígito ou letra)
        do avança;
    Retorna (identificador);

N:  While entrada é dígito
        do avança;
    Retorna (inteiro)

X:  Avança;
    Retorna (caracter especial);
```

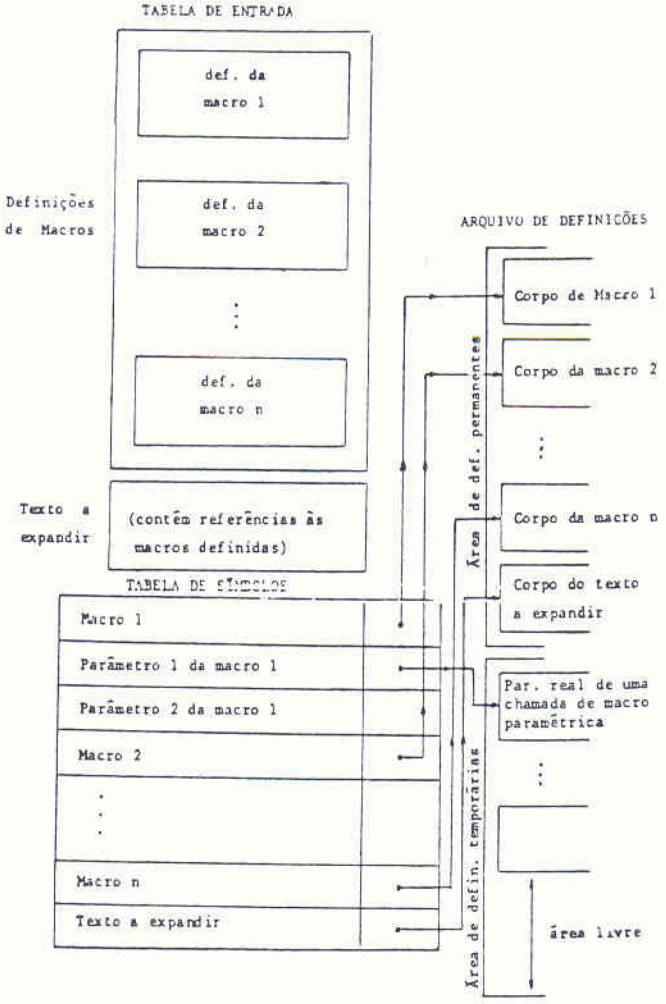


Figura 7 - Estruturas de Dados do Macro-Expansor