CENTRO UNIVERSITÁRIO FEEVALE

MÁRCIO MIGUEL GOMES

COMPILADOR APLICADO À EXTRAÇÃO DE DADOS DE ANÁLISES AMBIENTAIS

Novo Hamburgo, junho de 2007.

MÁRCIO MIGUEL GOMES

COMPILADOR APLICADO À EXTRAÇÃO DE DADOS DE ANÁLISES AMBIENTAIS

Centro Universitário Feevale
Instituto de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação
Trabalho de Conclusão de Curso

Professor Orientador: Ricardo Ferreira de Oliveira

Novo Hamburgo, junho de 2007.

AGRADECIMENTOS

A Deus pela vida e saúde, aos pais André e Felícita pelo amor, carinho e educação, aos irmãos Adriano, Regina e Henrique pelo companheirismo, aos mestres pelo conhecimento, aos amigos pela alegria e descontração, e a minha esposa Elisabeth, simplesmente por existir.

RESUMO

Os avanços da ciência permitiram o desenvolvimento de novas técnicas analíticas e isto se refletiu na evolução tecnológica dos equipamentos de análises laboratoriais. Juntamente com o crescimento da informática, criou-se um cenário propício para a automação da coleta de dados brutos, extração de informações e armazenamento de dados. Hoje não existe um padrão mundial para a disposição e armazenamento destes dados resultantes das análises laboratoriais, embora diversas normas e certificações exijam o armazenamento destes dados para questões de rastreabilidade. A automação na extração dos dados brutos e a padronização no armazenamento dos dados processados proporcionam inúmeras vantagens, como integridade, confidencialidade e rapidez. Isto reflete para o laboratório em forma de maior capacidade produtiva, menor quantidade de falhas operacionais e de transcrição de dados e principalmente em redução de custos, que resulta em uma preciosa vantagem competitiva. Este estudo tem por objetivo propor um modelo de compilador capaz de interpretar dados brutos de diversos equipamentos analíticos e armazená-los em banco de dados para uso futuro.

Palavras-chave: Aquisição de dados. Extração de dados. Automação de laboratórios. Compiladores.

ABSTRACT

The advances of science have allowed the development of new analytical techniques and this is reflected in the technological evolution of the analytical equipments. With the growth of computer science, it was created an ideal scene for the automation of raw data acquisition, information extraction and data storage. Nowadays, does not exist a world-wide standard for the presentation and storage of data from laboratories analyses, even so, a several norms and certifications demands the storage of these data for tracking reasons. The automation in the extraction of raw data and the standardization in the storage of the processed data provide a lot of advantages, as integrity, confidentiality and speed. This reflects for the laboratory as a bigger productive capacity, lesser amount of operational errors in data transcription and mainly in reduction of costs, which results in a precious competitive advantage. This research has as objective to consider a model of compiler capable to process raw data from diverse analytical equipments and to store them in data base for future use.

Keywords: Data Acquisition. Data Extraction. Laboratories automation. Compilers.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AFD	Autômato Finito Determinístico (AHO et al, 1995, p.52)
AFN	Autômato Finito Não-Determinístico (AHO et al, 1995, p.52)
AM	Ante Meridiem ou antes do meio dia
ASCII	American Standard Code for Information Interchange ou Código Padrão Americano Para Troca de Informações
BNF	Forma de Backus-Naur ou Forma Normal de Backus (GRUNE et al, 2001, p.34)
FDA	Food and Drug Administration ou Administração de Alimentos e Drogas
HTML	HyperText Markup Language ou Linguagem de Marcação de Hipertexto
ISO	International Standardization Organization ou Organização Internacional para Padronização
PC	Personal Computer ou Computador Pessoal
PM	Post Meridiem ou após o meio dia
RDC	Resolução da Diretoria Colegiada

SUMÁRIO

INTRODUÇÃO

O avanço do uso da informática em todas as áreas da sociedade nas últimas décadas também ocorreu para laboratórios de controle de qualidade. Equipamentos analíticos tecnologicamente ultrapassados e técnicas manuais foram gradativamente sendo substituídos por equipamentos atualizados e com novos recursos disponíveis.

Em decorrência de um mercado cada vez mais competitivo e da alta exigência de qualidade, muitos laboratórios tiveram nos últimos anos um incremento significativo do número de amostras e das análises realizadas e, com a pressão pela redução de custos, reduziram suas equipes ou as mantiveram igual no melhor dos casos.

Em função dos ambientes regulamentados por normas como a ISO 17025, RDC 210 e FDA, a carga de trabalho administrativo dos analistas aumentou drasticamente em função da necessidade de incremento de registros necessários para garantir e evidenciar a confiabilidade e rastreabilidade dos resultados gerados pelo laboratório.

O aumento da rigidez nestas normas que regem o controle de qualidade de laboratórios de análises juntamente com o aumento da concorrência no mercado impulsionaram a necessidade de informatização e automatização de processos.

Assim, as empresas investiram em novas técnicas analíticas ou equipamentos que tornaram o processo de análise mais rápido e confiável, resultando em aumento de qualidade, confiabilidade e conseqüentemente lucratividade dos laboratórios. Entretanto, na maioria dos casos, todo este volume de medições e dados gerados continua sendo coletado, processado, gerenciado e armazenado de forma manual.

Uma das maneiras de aumentar a confiabilidade e reduzir o tempo de processamento dos resultados de análises consiste em automatizar a leitura e transcrição dos resultados

encontrados durante a realização das análises. Processos até então manuais de transcrição de dados, com um enorme potencial de falha humana e desperdício de tempo podem ser substituídos por processos informatizados que reduzem o erro a níveis mínimos e elevam a produtividade a níveis nunca antes imaginados.

Os equipamentos analíticos comercializados atualmente possuem recursos que vão muito além da realização da análise propriamente dita. Uma grande quantidade de equipamentos é totalmente automatizada por algum *software* que roda em um PC externo ao equipamento analítico. Outros equipamentos possuem processadores ou controladores internos, mas disponibilizam formas de comunicação de dados com o ambiente externo.

Todos estes equipamentos possuem uma forma eletrônica de apresentação dos resultados obtidos no processo de análise. Normalmente equipamentos que possuem um PC e *software* de controle possuem também uma maneira de exportar os dados, como arquivos texto, por exemplo. Já equipamentos com controladores internos comumente disponibilizam seus dados por uma interface de comunicação serial ou paralela.

Os dados exportados pelos equipamentos na maioria dos casos não possuem uma estrutura simples, tampouco possuem padronização de formato entre fabricantes distintos. Também ocorre de equipamentos distintos de um mesmo fabricante apresentarem formatos de relatórios completamente diferentes. Mesmo assim, estes dados brutos são estruturados e podem perfeitamente ser interpretados por um *software* preparado especialmente para isso.

Para um *software* conseguir interpretar e extrair informações a partir destes relatórios ele dever ter a capacidade de reconhecer os padrões e as estruturas destes arquivos de dados. Ele deverá ser completo o bastante a ponto de perceber variações do padrão e estrutura, e mesmo assim conseguir interpretar os dados e extrair as informações desejadas.

GRUNE et al (2001, p.7) mostram que “As técnicas de construção de compiladores podem ser e são aplicadas fora da construção de compiladores em seu sentido mais estrito. [...] Os exemplos são a leitura de dados estruturados, a introdução rápida de novos formatos e os problemas gerais de conversão de arquivos”.

Assim, o objetivo principal deste trabalho concentra-se no estudo de compiladores para serem usados na interpretação de dados brutos e extração de informações, e não para

serem usados na sua maneira mais clássica, ou seja, leitura de uma linguagem de programação de alto nível e estruturada e geração de código de máquina executável.

A primeira seção deste trabalho registra o estudo teórico dos compiladores com o objetivo de adquirir conhecimento suficiente para certificar-se que é possível utilizar um compilador na extração de informações a partir de dados brutos. Este estudo também serve de base para a etapa complementar deste trabalho, que é a construção de um protótipo do compilador.

A segunda seção apresenta a realidade geral dos laboratórios de análises ambientais nas regiões Sul e Sudeste do Brasil, embasado em experiência própria durante nove anos, em trabalhos diários com automação de diversos processos em dezenas de laboratórios de controle de qualidade.

A terceira seção define um modelo de relatório de dados brutos a ser utilizado como base para o projeto e desenvolvimento do protótipo do compilador e segue com diversos estudos em cima de problemas reais de identificação de padrões e formas de extração de informações a partir dos dados brutos.

1 COMPILADORES

> "Em sua forma mais geral, um compilador é um programa que aceita como entrada um texto de programa em uma certa linguagem e produz como saída um texto de programa em outra linguagem, enquanto preserva o significado deste texto. Este processo é chamado de tradução, como seria denominado se os textos estivessem em linguagens naturais." (GRUNE et al, 2001, p.1).

Um compilador, segundo AHO et al (1995), é um programa que lê um programa em uma linguagem fonte e o traduz em uma outra linguagem alvo. (ver Figura 1.1)

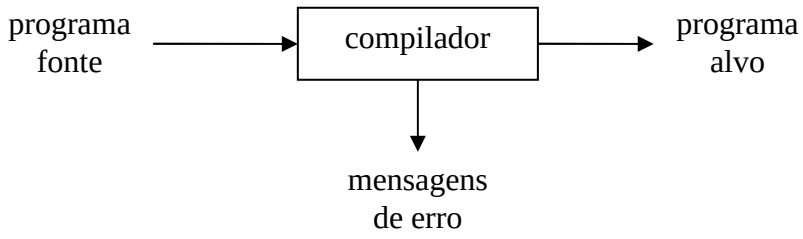

Figura 1.1 - Um compilador. (AHO et al, 1995, p.1).

A compilação pode ser dividida em duas grandes partes: análise e síntese. Segundo AHO et al (1995), a etapa de análise divide o programa fonte em pequenas partes e cria representações intermediárias para estas partes, enquanto a síntese constrói o programa alvo desejado a partir destas representações intermediárias.

Para RANGEL (1999), a sintaxe é associada à idéia de forma, em oposição à semântica, associada a significado e conteúdo. Assim, em princípio, a sintaxe de uma linguagem de programação deve descrever todos os aspectos relativos à forma de construção de programas corretos na linguagem, enquanto a semântica deve descrever o que acontece quando o programa é executado. Portanto, toda a análise está relacionada com sintaxe, e a semântica deveria corresponder apenas à geração de código, que deve preservar o significado do programa fonte, construindo um programa objeto com o mesmo significado.

1.1 Gramáticas

> “As gramáticas ou, mais precisamente gramáticas livres de contexto, constituem o formalismo essencial para descrever a estrutura de programas em uma linguagem de programação. Em princípio, a gramática de uma linguagem descreve apenas a estrutura sintática, mas como a semântica de uma linguagem é definida em termos de sintaxe, a gramática também é instrumental na definição da semântica.” (GRUNE et al, 2001, p.31).

RANGEL (1999) afirma que quase universalmente, a sintaxe das linguagens de programação é descrita por gramáticas livres de contexto, em uma notação chamada BNF (Forma de Backus-Naur ou ainda Forma Normal de Backus), ou em alguma variante ou extensão dessa notação.

Uma gramática consiste em um conjunto de regras de produção e um símbolo de partida. GRUNE (2001) estrutura uma gramática da seguinte forma:

- Cada regra de produção define uma construção sintática nomeada;

- Uma regra de produção consiste em duas partes: um lado esquerdo e um lado direito, separados por um caractere especial;

- O lado esquerdo é o nome da construção sintática e o lado direito mostra uma forma possível da construção sintática;

- O lado direito de uma regra de produção pode conter dois tipos de símbolos: terminais e não terminais. Símbolo terminal é um ponto final do processo de produção.

Um exemplo de regra de produção é:

expressão → '(' expressão operador expressão ')'

A Figura 1.2 mostra a forma de derivação da string (1*(1+1)).

```
expressão
'(' expressão operador expressão ')'
'(' '1' operador expressão ')'
'(' '1' '*' expressão ')'
'(' '1' '*' '(' expressão operador expressão ')' ')'
'(' '1' '*' '(' '1' operador expressão ')' ')'
'(' '1' '*' '(' '1' '+' expressão ')' ')'
'(' '1' '*' '(' '1' '+' '1' ')' ')'
```

Figura 1.2 - Derivação da string (1*(1+1)) (GRUNE et al, 2001, p.33).

1.2 Análise

A análise é a etapa da compilação em que é verificado se o programa fonte pertence a uma gramática, e é composta por 3 fases: léxica, sintática e semântica.

RANGEL (1999) afirma que a análise léxica tem como finalidade a separação e identificação dos elementos componentes do programa fonte e normalmente esses componentes são especificados através de expressões regulares. A análise sintática deve reconhecer a estrutura global do programa, descrita através de gramáticas livres de contexto. A análise semântica se encarrega da verificação das regras restantes. Essas regras tratam quase sempre da verificação de que os objetos são usados no programa da maneira prevista em suas declarações, por exemplo, verificando que não há erros de tipos.

1.2.1 Análise léxica

> "Após a análise léxica, itens como identificadores, operadores, delimitadores, palavras chave e palavras reservadas estão identificados normalmente através de duas informações: um código numérico e uma cadeia de símbolos. Estes itens são conhecidos como componentes léxicos do programa, lexemas, ou ainda tokens." (RANGEL, 1999).

A análise léxica é a primeira etapa de um compilador e serve para separar e classificar os elementos de um programa fonte. Ela também elimina elementos decorativos e de organização visual, como espaços, quebras de linha e comentários. Pode ser subdividida em duas tarefas: varredura ou *scanner* e análise léxica.

A varredura consiste em ler caractere por caractere do programa fonte e entregá-los ao analisador léxico, que por sua vez realiza a análise do conjunto de caracteres que recebeu do *scanner* para identificar lexemas conhecidos da gramática. Esta identificação pode ser feita através de uma técnica de autômatos finitos - também chamada de máquina de estados - que compara a seqüência de caracteres recebidos do *scanner* com expressões regulares. Assim, a expressão regular identifica o tipo de dado que a seqüência de caracteres representa.

Para RANGEL (1999), a implementação de reconhecedores de linguagens regulares pela técnica de autômatos finitos é mais simples e mais eficiente do que a implementação de reconhecedores de linguagens livres de contexto utilizando autômatos de pilha. Como é possível usar expressões regulares para descrever a estrutura de componentes básicos das linguagens de programação, tais como identificadores, palavras reservadas, literais numéricos,

operadores e delimitadores, a análise léxica é implementada separadamente pela simulação de autômatos finitos.

Para a linha de comando `montante := deposito + taxa_juros * 60` o analisador léxico encontraria os seguintes *tokens*:

Tabela 1.1 - Tokens encontrados pelo analisador léxico.

Tipo do token	Valor do token
Identificador	montante
Símbolo de atribuição	:=
Identificador	deposito
Operador de adição	+
Identificador	taxa_juros
Operador de multiplicação	*
Número	60

1.2.2 Expressões regulares

Expressões regulares são modelos matemáticos que definem quais caracteres são aceitos em um conjunto, qual critério de repetição e o posicionamento dos caracteres dentro do conjunto.

Uma expressão regular, segundo GRUNE et al (2001), é uma fórmula que descreve um conjunto de *strings* possivelmente infinito. Como uma gramática, ela pode ser vista tanto como uma receita para gerar estes *strings* quanto como um padrão para combiná-los.

Existem diversas operações que podem ser aplicadas a expressões regulares, mas conforme AHO et al (1995), para a análise léxica são utilizadas as expressões de união, concatenação e fechamento, conforme Tabela 1.2.

Tabela 1.2 - Definições de operações em linguagens (AHO et al, 1995, p.43).

Operação	Definição
união de *L* e *M*, escrita $L \cup M$	$L \cup M = \{s \mid s$ está em L ou s está em $M\}$
concatenação de *L* e *M*, escrita *LM*	$LM = \{st \mid s$ está em L e st está em $M\}$
fechamento de Kleene de L, escrito L*	$L^* = \bigcup_{i=0}^{\infty} L^i$ *L** denota “zero ou mais concatenações de” *L*.
Fechamento positivo de L, escrito L^+	$L^+ = \bigcup_{i=1}^{\infty} L^i$ *L** denota “uma ou mais concatenações de” *L*.

Desta forma é possível verificar se um conjunto de caracteres lidos pelo *scanner* pode ser gerado por determinada expressão regular ou não, e classificar tal conjunto de caracteres com sendo um identificador, operador, símbolo de atribuição, número etc.

1.2.3 Diagramas de transições

O diagrama de transições é a representação gráfica (Figura 1.3) de uma máquina de estados. As posições são representadas por círculos e são chamadas de *estados*. “Os estados são conectados por setas chamados de *lados*. Os lados que deixam o estado *s* possuem rótulos indicando os caracteres de entrada que podem aparecer após o diagrama de transições ter atingido o estado *s*. O rótulo *outro* se refere a qualquer caractere que não seja indicado por qualquer um dos lados que deixam *s*”, dizem AHO et al (1995, p.45).

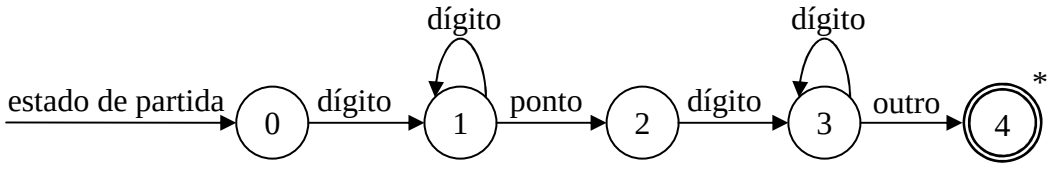

Figura 1.3 - Diagrama de transição para número (AHO et al, 1995, p.47).

1.2.4 Autômatos finitos

Autômato finito é o modelo matemático de uma máquina de estados finitos. Pode ser representado por um diagrama de estados conforme Figura 1.3, onde se definem os estados, transições e ações.

Um autômato finito pode ser determinístico ou não determinístico, onde “não determinístico” significa que mais de uma transição para fora de um estado pode ser possível para um mesmo símbolo de entrada, afirmam AHO et al (1995, p.51).

1.2.4.1 Autômatos finitos não-determinísticos

Um *autômato finito não-determinístico* (AFN) é um modelo matemático que, segundo AHO et al (1995), consiste em:

um conjunto de estados *s*

um conjunto de símbolos de entrada Σ

uma função de transição, movimento, que mapeia pares *estado-símbolo* em conjunto de estados

um estado s_0 que é distinguido como o *estado de partida*

um conjunto de estados *F* distinguidos como *estados de aceitação*

A particularidade do AFN é que o mesmo caractere pode rotular duas ou mais transições para fora de um mesmo estado e os lados podem ser rotulados pelo símbolo especial $\in$ bem como pelos símbolos de entrada. O símbolo $\in$ representa cadeias de caracteres vazias.

O diagrama da Figura 1.4 mostra um AFN que aceita, por exemplo, as cadeias de entrada *abb*, *aabb*, *babb*, *aaabb*. Este autômato aceita qualquer combinação das letras *a* e *b*, desde que finalizada pela seqüência *abb*.

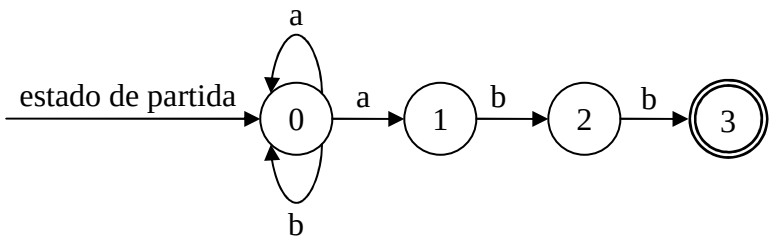

Figura 1.4 - Um autômato finito não-determinístico (AHO et al, 1995, p.52).

1.2.4.2 Autômatos finitos determinísticos

Um autômato *finito determinístico* (AFD) é um caso especial de autômato finito não-determinístico, no qual:

nenhum estado possui uma transição-$\in$**,** isto é, uma transição à entrada $\in$**,** e;

para cada estado *s* e símbolo de entrada *a* existe no máximo *um* lado rotulado *a* deixando *s*;

"Um autômato finito possui no máximo uma transição, a partir de cada estado, para qualquer símbolo de entrada. Como conseqüência, é muito fácil determinar se um AFD aceita uma cadeia de entrada, dado que existe no máximo um único percurso, rotulado por aquela cadeia, a partir do estado inicial." (AHO et al, 1995, p.53).

A Figura 1.5 mostra a representação de um AFD que corresponde ao AFN da Figura 1.4.

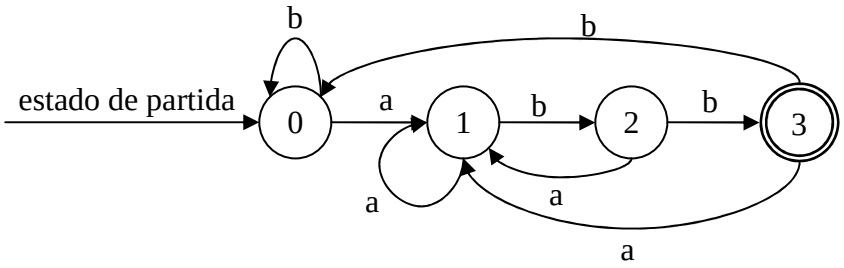

Figura 1.5 - Um autômato finito determinístico (AHO et al, 1995, p.53).

1.2.5 Análise sintática

A análise sintática, ou *parser*, é a segunda etapa de um compilador. O analisador sintático obtém uma cadeia de *tokens* provenientes do analisador léxico e verifica se a mesma pode ser gerada pela gramática da linguagem fonte, segundo AHO et al (1995).

A função do analisador sintático é verificar se as construções usadas no programa estão gramaticalmente corretas. Com a gramática correta, o analisador sintático busca descobrir formas de derivação desta gramática. (NETO, 1987)

A análise sintática deve reconhecer a estrutura global do programa, por exemplo, verificando que programas, comandos, declarações, expressões etc têm as regras de composição respeitadas.

O analisador sintático, ao interpretar o código

```
if x > 0 then modx := x else modx := (-x)
```

deve identificar que o texto se trata de um `<comando>`, no caso um `<comando-if>`, composto pela palavra reservada `if`, seguida de uma `<expressão>`, seguida da palavra reservada `then` etc. Os itens `<expressão>` e `<atribuição>` ainda podem ser decompostos em fragmentos menores, conforme Figura 1.6. (RANGEL, 1999).

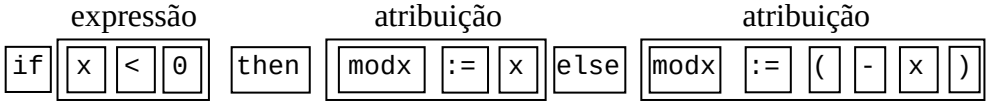

Figura 1.6 - Comando-If (RANGEL, 1999).

Para RANGEL (1999), os métodos de análise sintática podem ser classificados segundo a maneira pela qual a árvore de derivação da cadeia analisada x é construída:

- nos métodos descendentes, a árvore de derivação correspondente a x é construída de cima para baixo, ou seja, da raiz (o símbolo inicial S) para as folhas, onde se encontra x.

- nos métodos ascendentes, a árvore de derivação correspondente a x é construída de baixo para cima, ou seja, das folhas, onde se encontra x, para a raiz, onde se encontra o símbolo inicial S.

1.2.5.1 Análise sintática descendente top-down

A análise sintática *top-down* pode ser vista como uma tentativa de se encontrar uma derivação mais a esquerda para uma cadeia de entrada. Equivalentemente, pode ser vista como uma tentativa de se construir uma árvore gramatical, para a cadeia de entrada, a partir da raiz, criando os nós da árvore gramatical em pré-ordem. (AHO et al, 1995).

Para GRUNE et al (2001), um analisador sintático *top-down* começa construindo o nó superior da árvore, e segue construindo os demais nós da árvore sintática em pré-ordem, o que significa que a parte superior de uma árvore é construída antes de qualquer um de seus nós inferiores. A Figura 1.7 ilustra a seqüência de construção dos nós em uma análise sintática descendente.

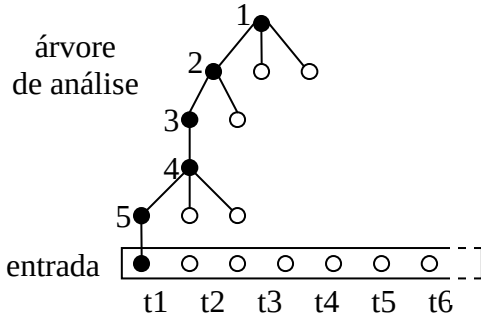

Figura 1.7 - Um analisador *top-down* (GRUNE et al, 2001, p.104).

1.2.5.2 Análise sintática ascendente bottom-up

O método de análise *bottom-up* constrói os nós da árvore sintática em pós-ordem. GRUNE et al (2001) afirmam que a parte superior de uma subárvore é construída após todos os seus nós inferiores terem sido construídos. Quando um analisador *bottom-up* constrói um

nó, todos os seus filhos já foram construídos, estão presentes e são conhecidos. O rótulo do próprio nó também é conhecido. Em seguida o analisador cria o nó, o identifica e o conecta a seus filhos. A Figura 1.8 ilustra a seqüência de construção dos nós em uma análise sintática ascendente.

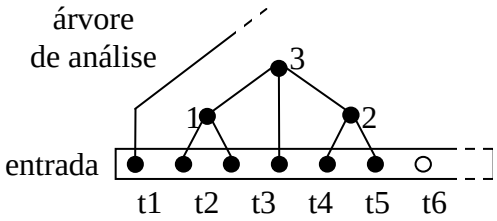

Figura 1.8 - Um analisador *bottom-up* (GRUNE et al, 2001, p.105).

1.2.6 Análise semântica

Análise semântica é a terceira fase da compilação onde se verificam os erros semânticos no código-fonte, por exemplo, uma multiplicação entre tipos de dados diferentes. A análise semântica também coleta as informações necessárias para a fase seguinte da compilação, que é a geração de código-fonte.

Fundamentalmente, a análise semântica trata os aspectos sensíveis ao contexto da sintaxe das linguagens de programação. RANGEL (1999) dá como exemplo que não é possível representar em uma gramática livre de contexto uma regra do tipo: "Todo identificador deve ser declarado antes de ser usado.". A verificação de que essa regra foi aplicada cabe à análise semântica.

> "... o tratamento de contexto se preocupa com relações de longa distância. Por exemplo, ele relaciona o tipo de variável em uma instrução a seu uso em uma expressão, e relaciona a posição de um rótulo a seu uso em uma instrução *goto*. Os conectores nessas relações de longo alcance são os identificadores. Todas as ocorrências aplicadas de um identificador *i* em uma expressão são plugues que se encaixam em um soquete - a declaração para *i* - e desse soquete eles obtêm informações sobre o tipo, a duração e assim por diante." (GRUNE et al, 2001, p.400).

Além da verificação de tipos de dados, o analisador semântico é responsável por perceber as falhas em fluxos de controle e unicidade de comandos.

Uma possibilidade de falha de tipos de dados ocorre quando membros de uma expressão possuem tipos de dados incompatíveis, por exemplo, uma operação de soma não pode conter um elemento de texto, apenas elementos numéricos.

> "Um importante componente da análise semântica é a verificação de tipos. Nela, o compilador checa se cada operador recebe os operandos que são permitidos pela especificação da linguagem fonte. Por exemplo, muitas definições nas linguagens de programação requerem que o compilador relate um erro a cada vez que um número real seja usado para indexar um *array*. No entanto, a especificação da linguagem pode permitir algumas coerções de operandos, como, por exemplo, quando um operador aritmético binário é aplicado a um inteiro e a um real. Nesse caso, o compilador pode precisar converter o inteiro para real". (AHO et al, 1995, p.4).

Uma falha em fluxo de controle ocorre quando uma instrução de quebra de seqüência de comandos é declarada fora de um comando de repetição, conforme Figura 1.9. Por exemplo, na linguagem Pascal o comando de quebra de execução *break* obrigatoriamente deve ser usado dentro de um bloco de repetição, como o *for* ou *while*. Ver Figura 1.10.

```
for i := 2 to 10 do
  x := x * i;

if x > 1000 then
  break               // uso incorreto da instrução break
else
  print(x);
```

Figura 1.9 - Uso incorreto do comando *break.*

```
for i := 2 to 10 do
  begin
    x := x * i;
    if x > 1000 then
      break           // uso correto da instrução break
    else
      print(x);
  end;
```

Figura 1.10 - Uso correto do comando *break.*

Uma falha em unicidade de comandos pode ocorrer quando um elemento somente pode ser declarado uma única vez em. Um bom exemplo de falha de unicidade ocorre quando dentro de uma instrução *case* na linguagem Pascal é declarado mais de uma vez o mesmo rótulo, conforme Figura 1.11.

```
case i of
  1: ExecutaTarefaUm;
  1: ExecutaTarefaUm;   // falha de unicidade de comandos
  2: ExecutaTarefaDois;
  3: ExecutaTarefaTres;
else
  ExecutaTarefaExcecao;
end;
```

Figura 1.11 - Falha em unicidade de comandos.

2 SITUAÇÃO ATUAL DOS LABORATÓRIOS DE ANÁLISE

Um laboratório de análises ambientais realiza basicamente análises em amostras de águas, efluentes, solos e emissões atmosféricas. As amostras analisadas representam muitas vezes um dejeto resultante de um processo industrial a ser tratado antes de ser liberado para a natureza. Tais análises têm por objetivo verificar se os parâmetros analisados estão de acordo com as muitas legislações em vigor. Este estudo analítico auxilia na escolha do tratamento a ser aplicado no dejeto para adequação às normas ambientais, define se o resíduo pode ser incinerado ou não e ainda pode classificar o resíduo quanto à forma de armazenamento em um depósito de lixo contaminado.

2.1 Cenário dos laboratórios

Estudos realizados em diversos laboratórios de ensaios e controle de qualidade nas regiões Sul e Sudeste do Brasil mostram que os administradores e gerentes destes laboratórios desconhecem ou ignoram um excelente recurso que já possuem a disposição, capaz de aumentar a qualidade e confiabilidade das análises realizadas, reduzir drasticamente o tempo de permanência de uma amostra dentro do laboratório e aumentar a capacidade produtiva sem a necessidade de aquisição de novos equipamentos ou contratação de funcionários. Este recurso é a comunicação de dados entre equipamentos analíticos e o sistema de gerenciamento de informações do laboratório.

Quase que a totalidade dos laboratórios de ensaios e controle de qualidade possui equipamentos analíticos digitais, com geração de relatórios em arquivos texto ou exportação de dados eletrônicos. Porém, este importantíssimo recurso é simplesmente ignorado pela maioria das empresas, fazendo com que muitas horas de trabalho sejam literalmente desperdiçadas em transcrição, digitação e conferência de resultados.

Através de estudos e trabalhos realizados durante nove anos em dezenas de laboratórios em diversos estados brasileiros, chegou-se aos seguintes dados:

O trabalho administrativo do laboratório gasto em registro de informações para rastreabilidade, transcrição, conferência de dados e elaboração de relatórios, dependendo do tipo de laboratório e matrizes trabalhadas, toma até 60% do tempo total dos analistas;

Tipicamente o volume de dados e informações que saem do laboratório para outras áreas ou para os clientes em forma de resultados finais, é de no máximo 10% do volume total de dados que são coletados, tratados e registrados dentro de um laboratório;

Com este levantamento, consegue-se ter uma boa visão da quantidade de dados brutos que são gerados durante uma etapa analítica e quantos destes dados são realmente utilizados no final do processo.

Todos estes dados são necessários por questões de rastreabilidade e não podem ser eliminados, porém é possível melhorar os processos através da informatização e coleta e dados para reduzir a intervenção humana, e conseqüentemente liberar o tempo dos analistas para trabalhos mais nobres ou que exijam mais qualificação.

2.2 Realização de análises

Para a realização das análises nas amostras coletadas, são utilizadas diversas técnicas e equipamentos distintos. Cada tipo de amostra ou matriz (água, solo, ar) necessita de uma metodologia de trabalho diferente e equipamentos analíticos diferentes.

Entre os equipamentos analíticos de um laboratório existem desde vidrarias (balão volumétrico, copo de Becker, pipeta volumétrica, bureta etc) até equipamentos eletrônicos muito sofisticados, como espectrômetros, cromatógrafos e fotômetros.

As análises feitas através de vidrarias são totalmente manuais e a leitura dos resultados é feita quase sempre por método visual. Por exemplo, visualiza-se o volume de um líquido em uma escala ou se lê a temperatura de um fluido em um termômetro analógico de mercúrio. Depois se anota as medições em uma ficha de análise para posterior uso em cálculos e digitação em uma planilha eletrônica ou editor de textos.

Já os equipamentos mais modernos possuem exportação de dados em formato texto ou comunicação serial ou paralela. Nestes casos, a captura dos dados em meio eletrônico é possível, e a interpretação e extração de informações destes dados brutos dependem apenas de se programar rotinas capazes de executarem tal tarefa.

Porém, alguns equipamentos eletrônicos são muito antigos e não possuem interfaces para exportação de dados. Outros possuem tal interface, mas utilizam conectores fora do padrão. Também existem casos em que os dados exportados estão dispostos em formatos desorganizados ou binários, complicando muito sua extração. Raros são os equipamentos que possuem uma interface de exportação de dados padronizada e documentada, o que facilitaria muito a leitura das medições por um sistema externo.

2.3 Transcrição de dados

Uma das tarefas mais importantes que ocorrem em um laboratório é a transcrição dos dados obtidos durante as análises. Para laboratórios não informatizados ou parcialmente informatizados, o procedimento de realização da análise em uma amostra é sempre acompanhado da ficha de análise, onde o analista registra manualmente com caneta as medições que vai realizando.

Após o registro manual das medições, a ficha de análise é digitada em uma planilha eletrônica ou em muitos casos, são feitos cálculos manuais para obtenção de resultados de medições indiretas antes da digitação.

Por questões de tempo e custo, este processo de digitação e cálculo nem sempre é feito pela própria pessoa que realizou a análise. Considerando-se a possibilidade de erro humano na leitura e interpretação dos valores escritos manualmente na ficha de coleta e a possibilidade de erro de digitação, o processo de transcrição de resultados além de lento é muito susceptível a falhas.

Se pensarmos que um dado pode ter sido interpretado ou digitado de maneira errada por uma pessoa em um sistema informatizado, o trabalho do analista foi em vão, pois o resultado final da análise realizada que consta no relatório final da amostra não condiz com a realidade, por uma simples falha humana.

Como exemplo de tempo gasto em transcrição de resultados por mês para apenas um equipamento, pode-se usar a transcrição manual de resultados de análises de um equipamento de Raio X com média de 50 amostras por dia e avaliação típica de 20 metais por amostra.

Tipicamente toma-se de 3 a 4 segundos para a leitura visual e digitação de cada um dos valores da ficha de coleta, com cerca de 4 algarismos cada. Contabilizando 50 amostras, 20 metais e 3,5 segundos por digitação, obtém-se 58 minutos por dia ou cerca de 21 horas por mês apenas com o trabalho de digitação.

Somando a isso o trabalho prévio de anotação das medições na ficha de análise e a revisão dos dados digitados, chega-se seguramente ao dobro do tempo, ou seja, 42 horas por mês em um trabalho que pode ser substituído com grandes vantagens pela automação na coleta de dados.

2.4 Exportação de dados

Os equipamentos analíticos possuem basicamente dois tipos de interfaceamento: exportação de dados através de porta de comunicação (serial ou paralela) ou geração de arquivo texto com caracteres ASCII. No caso da comunicação de dados, geralmente os equipamentos trabalham com um protocolo unidirecional através de uma porta serial no padrão RS232C (TORRES, 2001). Alguns equipamentos possuem porta paralela. Neste caso, a finalidade da porta é transmitir dados para uma impressora, mas é possível capturar os dados e interpretá-los. É raro existir um protocolo de comunicação bidirecional onde o sistema externo precisa trocar dados com o equipamento analítico.

No caso de geração de arquivo texto, o equipamento analítico já possui um PC com um *software* de controle (chamado de *Workstation*), e este *software* se encarrega de controlar a realização das análises e gerar o arquivo texto em disco. É comum nos equipamentos atuais este PC ser uma máquina externa, padrão de mercado, conectada ao equipamento via um cabo específico e uma placa ligada a um barramento interno deste PC. Neste caso fica fácil instalar uma placa de rede e possibilitar o acesso aos dados remotamente. Porém em equipamentos mais antigos, o PC pode ser interno e fazer parte da máquina, podendo até possuir *hardware* específico do fabricante. É muito comum nestes casos não existir interface de rede, o que complica a coleta de dados.

Para os casos em que os equipamentos geram relatórios em arquivos texto, um *software* pode ser desenvolvido para monitorar uma determinada pasta ou subpastas a procura destes arquivos gerados. Ao encontrar um arquivo, o *software* pode então processá-lo e extrair as informações de interesse para a continuidade do trabalho analítico.

Já para equipamentos que transmitem os resultados através de comunicação de dados serial ou paralela, o *software* deve abrir um canal de comunicação e ficar na escuta da porta, guardando em um *buffer* os dados que chegam para posterior processamento e extração de informações.

A aquisição dos dados diretamente dos equipamentos para um banco de dados centralizado agiliza todo o processo analítico, pois é possível reduzir significativamente o tempo de leitura de resultados, anula o risco de erro humano na transcrição de dados, centraliza em um único local os dados de análises realizadas além de padronizar a forma com que os mesmos são armazenados. Isto permite que seja mantido o histórico de análises realizadas durante muitos anos por questões de rastreabilidade e permite que diferentes sistemas busquem nesta fonte de dados centralizada as informações que precisarem.

2.5 Proposta de solução

> "As técnicas de construção de compiladores podem ser e são aplicadas fora da construção de compiladores em seu sentido mais estrito. Como uma alternativa, outras formas de programação podem ser consideradas construção de compiladores, mais do que se consideraria tradicionalmente. Os exemplos são a leitura de dados estruturados, a introdução rápida de novos formatos e os problemas gerais de conversão de arquivos."
>
> "Se os dados têm uma estrutura clara, em geral é possível escrever uma gramática para eles. Utilizando-se um gerador de analisador, pode-se então gerar automaticamente um analisador. Por exemplo, tais técnicas podem ser aplicadas para criar rapidamente rotinas 'de leitura' para arquivos HTML, arquivos Postscript etc." (GRUNE et al, 2001, p.7-8).

A solução proposta é desenvolver um compilador capaz de reconhecer padrões gramaticais em dados brutos gerados por equipamentos analíticos e extrair as informações das medições realizadas.

Como cada conjunto de dados brutos vindo de equipamentos distintos provavelmente terá estrutura gramatical única, a solução prevê um compilador flexível e ajustável para reconhecer as diversas estruturas de dados e conseguir realizar sua interpretação e extração.

Após a extração, os dados serão gravados em banco de dados para uso futuro, como consultas, cálculos, comparativos com legislações etc. A estrutura da tabela de banco de dados utilizada para armazenar as informações extraídas dos dados brutos pode ser vista na Tabela 2.1.

Tabela 2.1 - Estrutura da tabela para armazenamento de dados.

Nome do campo	Tipo de dado	Descrição
AMOSTRA	Texto (50)	Identificação da amostra analisada
IDMEDICAO	Texto (100)	Identificação da medição
VLMEDICAO	Texto (250)	Valor da medição
UNMEDICAO	Texto (50)	Unidade da medição
DILMEDICAO	Fracionário	Diluição da amostra
DTMEDICAO	Data e Hora	Data e hora de realização da análise
RESPONSÁVEL	Texto (50)	Identificação do responsável pela análise
EQUIPAMENTO	Texto (100)	Equipamento utilizado na análise

3 LEVANTAMENTO DE DADOS BRUTOS

Devido a enorme variedade de equipamentos analíticos e relatórios de dados em formatos diversos, torna-se necessário escolher um modelo de relatório para dar continuidade a este trabalho. Com este relatório definido, se iniciam os estudos das estruturas de dados e padrões, para que sejam definidas as regras gramaticais que o compilador utilizará na extração das informações.

Um tipo de equipamento muito comum em laboratórios de análises ambientais é o espectrômetro gasoso (GCMS). Ele é utilizado para detectar concentrações de certas moléculas em uma amostra. O relatório final que um espectrômetro gera ao analisar uma amostra consiste basicamente em um cabeçalho com a identificação da amostra analisada, data e hora da análise, fator de diluição da amostra e identificação do arquivo de dados. Após o cabeçalho segue uma listagem com os nomes dos compostos encontrados e suas respectivas concentrações, seguidas da unidade de medição utilizada.

3.1 Escolha de um modelo de relatório

Para o desenvolvimento deste projeto, será utilizado como modelo o relatório de dados brutos de um espectrômetro gasoso GCMS que utiliza o *software* Chromquest da empresa Thermo. Para exportação dos dados é utilizado o *software* XCalibur. Os dados brutos de um espectrômetro gasoso GCMS exportados pelo XCalibur estão listados na Figura 3.1.

O conjunto de dados contidos no relatório pode variar muito. No cabeçalho, além dos dados citados anteriormente, podem ainda constar nome do método utilizado para leitura dos compostos, nome do operador do equipamento, identificação do equipamento utilizado e até informações diversas configuráveis pelo próprio operador. Na listagem dos compostos encontrados na amostra pode conter o número seqüencial do composto encontrado, o tempo

de retenção do composto dentro do equipamento, o tempo de retenção esperado e uma infinidade de informações muito específicas da área química analítica.

Além da vasta combinação de dados possíveis nos relatórios de um equipamento, ainda existe a variação de formatos de relatórios por versões diferentes dos *softwares* de controle dos equipamentos analíticos e ainda mais diferenças entre equipamentos de diferentes fabricantes.

```
Data File:              MS124371
Original Data Path:     MS124371.RAW
Sample Name:            22548
Acquisition Date:       01/18/07 16:50:19
Dilution Factor:        1.00
Instrument Method:      C:\Xcalibur\methods\pah_sim.meth

Name
Calculated Amount
Units
Acenafteno
5.282
PPB
2-Fluorbifenil
437.365
PPB
Fluoreno
N/A
PPB
Acenaftileno
0.925
PPB
Criseno
N/A
PPB
Fluoranteno
0.935
PPB
D10-Acenafteno
N/A
ppb
Naftaleno
26.073
PPB
D14-Terfenil
900.374
PPB
D12-Perileno
N/A
ppb
Benzo(k)fluoranteno
N/A
PPB
```

Figura 3.1 - Dados brutos de um espectrômetro gasoso GCMS.

3.2 Identificação dos dados

Neste exemplo da Figura 3.1 é possível distinguir o cabeçalho do relatório entre os textos `Data File` e `Instrument Method` e o conjunto de dados analisados a partir do texto `Name`.

Desta forma, o compilador a ser desenvolvido para extração dos dados deve identificar cada um dos textos `Data File`, `Sample Name`, `Acquisition Date`, `Dilution Factor` e `Instrument Method`, e extrair os dados referentes a estes identificadores. Também tem que ser capaz de localizar o bloco com a seqüência de compostos analisados, para extração do nome do composto, concentração e unidade de medição.

3.3 Problemas na identificação de dados e padrões

Conforme citado anteriormente, os relatórios de dados gerados pelos equipamentos não necessariamente são idênticos em formato. Dependendo do equipamento, versão do *software* de controle ou configuração utilizada, os formatos dos dados podem ser diferentes para equipamentos e métodos analíticos iguais.

Por isso, deve-se definir uma gramática flexível e projetar um compilador capaz de perceber tais diferenças e tratá-las de forma a extrair os dados corretamente.

3.3.1 Informações não constantes

Algumas informações não são essenciais para realização da análise e podem opcionalmente constar no relatório exportado. Em muitos dos casos, estas informações são apenas informativas ou observações escritas pelo analista no momento da execução da análise. Desta forma, informações que não aparecem em todos os relatórios gerados devem ser previstas como opcionais nas regras sintáticas.

A Figura 3.2 mostra um cabeçalho completo de um relatório de dados brutos de um espectrômetro GCMS. Já a Figura 3.3 mostra um relatório semelhante do mesmo equipamento, porém sem as informações `Original Data Path` e `Instrument Method`.

```
Data File:              MS124371
Original Data Path:     MS124371.RAW
Sample Name:            22548
Acquisition Date:       01/18/07 16:50:19
Dilution Factor:        1.00
Instrument Method:      C:\Xcalibur\methods\pah_sim.meth
```

Figura 3.2 - Cabeçalho completo de um espectrômetro GCMS.

```
Data File:              FID45754
Sample Name:            37844
Acquisition Date:       01/18/07 16:50:19
Dilution Factor:        1.00
```

Figura 3.3 - Cabeçalho incompleto de um espectrômetro GCMS.

3.3.2 Informações com formatos distintos

Os problemas mais comuns de formatos distintos para uma mesma informação consistem em definição do caractere separador decimal para tipos de dados numéricos e a ordem e quantidade de dígitos para expressar datas e horas.

O problema do caractere separador decimal é relativamente simples de resolver, pois no momento da análise léxica, é possível assumir como separador decimal tanto o ponto quanto a vírgula. Neste caso, um exemplo de autômato finito que represente um número fracionário com separador decimal ponto ou vírgula é ilustrado na Figura 3.4.

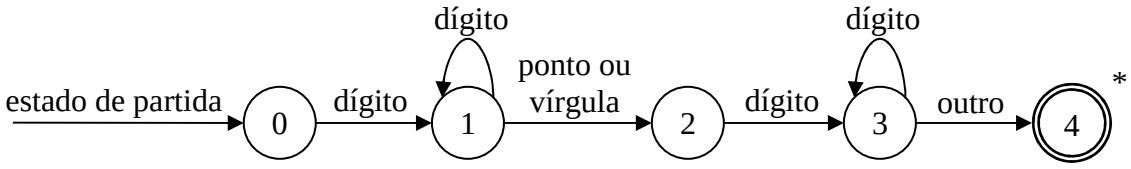

Figura 3.4 - Autômato finito de número com separador decimal ponto ou vírgula.

Já o problema de formatação de data e hora é bem mais complexo, pois este tipo de dado pode aparecer nos formatos mais distintos. A Tabela 3.1 apresenta o formato ideal para dados que representam data e hora, e a Tabela 3.2 mostra um formato de data e hora que exige uma interpretação distinta e suposição dos dois algarismos de milhar e centena do ano.

Tabela 3.1 - Data no formato dd/mm/aaaa hh:nn:ss.

Ordem	Dado	Formato
1	dia	2 algarismos
2	separador	barra
3	mês	2 algarismos
4	separador	barra
5	ano	4 algarismos
6	separador	espaço
7	hora	2 algarismos
8	separador	dois pontos
9	minuto	2 algarismos
10	separador	dois pontos
11	segundo	2 algarismos

Tabela 3.2 - Data no formato m/d/aa hh:nn:ss.

Ordem	Dado	Formato
1	mês	1 ou 2 algarismos
2	separador	barra
3	dia	1 ou 2 algarismos
4	separador	barra
5	ano	2 algarismos
6	separador	espaço
7	hora	2 algarismos
8	separador	dois pontos
9	minuto	2 algarismos
10	separador	dois pontos
11	segundo	2 algarismos

Além destes formatos ilustrados acima, verificou-se que é possível ocorrer também tipos de dados data e hora com o mês expresso pelas três primeiras letras do nome do mês e hora com 12 horas seguida do texto “AM” ou “PM”, que expressa se é antes ou após o meio dia.

Outro exemplo clássico de dados com formatos distintos ocorre quando uma análise não pode ser realizada ou apenas um elemento da análise não foi analisado por motivos diversos. Neste caso, é muito comum constar como resultado da análise não um número fracionário, mas o texto “N.A.”, “N/A”, “NA” ou derivações dele, que significam “Não Analisado”. Em outros casos, uma análise não realizada pode simplesmente não retornar valor algum. A Figura 3.5 ilustra um relatório que apresenta elementos que não puderam ser analisados, que são o `Estireno`, o `Bromobenzeno` entre outros.

```
Name
Calculated Amount
trans-1,2-Dicloroeteno
1.740
Estireno
N/A
Diclorodifluorometano
2.444
Bromobenzeno
N/A
Bromoclorometano
6.143
Butilbenzeno
N/A
Ethylbenzene
N/A
Clorometano
2.009
```

Figura 3.5 - Exemplo de relatório com elementos não analisados.

Uma variação desta característica ocorre quando o valor de concentração encontrado para um elemento é muito baixo, fora do limite mínimo confiável de detecção do equipamento analítico ou muito alto, além da capacidade de detecção.

Analogamente, em uma régua escolar que possui escala milimétrica e tamanho de 30 centímetros, é impossível medir com ela dimensões inferiores a 1 milímetro e superiores a 30 centímetros com precisão. Da mesma forma, um espectrômetro possui um limite inferior e um limite superior para medições. Quando um valor medido é inferior ao limite de detecção, pode ocorrer de o valor ser expresso com o algarismo zero ou os textos “N.D.”, “N/D”, “ND” ou derivações, que significam “Não Detectado”.

3.3.3 Tokens semelhantes sensíveis ao contexto

Pode ocorrer de palavras iguais terem significados distintos em um relatório de dados brutos. Uma palavra pode ter um significado quando está no cabeçalho e outro significado quando está no conjunto de dados. A Figura 3.6 mostra que a palavra “`Name`” antecedida pela palavra “`Sample`” no cabeçalho serve para identificar o código da amostra analisada, enquanto na seqüência, a mesma palavra “`Name`” tem o significado de iniciar o bloco de resultados dos compostos analisados na amostra.

```
Sample Name:            22548
Acquisition Date:       01/18/07 16:50:19
Dilution Factor:        1.00
Instrument Method:      C:\Xcalibur\methods\pah_sim.meth

Name
Calculated Amount
Units
Acenafteno
5.282
PPB
```

Figura 3.6 - Palavras iguais em contextos diferentes.

3.3.4 Remoção de sujeira

Algumas informações devem ser ignoradas do conjunto de dados brutos, pois são apenas decorativas ou não têm valor como medição analítica. Espaços e linhas em branco devem ser considerados como opcionais na definição das regras gramaticais para não interferirem na extração das informações.

Na Figura 3.7 é possível perceber os textos “`Name`”, “`Calculated Amount`” e “`Units`”. Apenas o texto “`Name`” é necessário para a definição do início do bloco que contém os resultados da análise realizada. Os demais textos são apenas decorativos e não devem ser extraídos do conjunto de dados brutos.

```
Name
Calculated Amount
Units
Acenafteno
5.282
PPB
2-Fluorbifenil
437.365
PPB
```

Figura 3.7 - Remoção de sujeira dos dados brutos.

CONCLUSÃO

O objetivo principal deste trabalho concentrou-se no estudo de compiladores para serem usados na interpretação de dados brutos e extração de informações, e não para serem usados na sua maneira mais usual, ou seja, geradores de códigos de máquinas.

Este estudo confirma a possibilidade de se desenvolver um compilador aplicado à extração de informações a partir de um conjunto de dados brutos, embora o uso clássico de compiladores consiste em leitura de uma linguagem de programação de alto nível e estruturada e geração de código de máquina executável.

O estudo sobre rotinas de laboratórios de análises ambientais e a detecção da necessidade de automatização da leitura de resultados analíticos ocorreu durante nove anos, em trabalhos diários com automação de diversos processos em dezenas de laboratórios de controle de qualidade nas regiões sul e sudeste do Brasil.

A utilização de técnicas de compiladores para a extração de informações a partir de dados brutos é plenamente viável, pois permite desenvolver uma solução elegante e de alto nível com relativa baixa complexidade para um problema muito comum em laboratórios de análises ambientais.

Este estudo será continuado com o objetivo de desenvolver o protótipo do compilador que irá extrair as informações dos dados brutos contidos em um relatório de análises gerado por um espectrômetro GCMS.

REFERÊNCIAS BIBLIOGRÁFICAS

AHO, Alfred V. Aho; SETHI, Ravi; ULLMAN, Jeffrey D. **COMPILADORES: princípios, técnicas e ferramentas.** Rio de Janeiro: LTC, 1995.

FISCHER, Charles N.; LEBLANC, Richard J. **Crafting a compiler with C.** California: Benjamin Cummings, 1991..

GRUNE, Dick et. al. Projeto moderno de compiladores: implementação e aplicações. Rio de Janeiro: Campus, 2001.

NETO, João J. **Introdução à compilação.** Rio de Janeiro: LTC, 1987.

PITTMAN, Thomas; PETERS, James. **The art of compiler design: theory and practice.** New Jersey: Prentice-hall, 1992.

RANGEL, José Lucas. **Material didático relativo à disciplina de Compiladores.** 1999. Disponível em: <http://www-di.inf.puc-rio.br/~rangel/comp.html> Acesso em: 12 de maio de 2007.

SEBESTA, Robert W. **Conceitos de linguagens de programação**. Porto Alegre: Bookman, 2000.

TORRES, Gabriel. **Hardware: curso completo** - 4. ed. 2001.