MAC 425/5739 — Decomposição em Fatores Primos Usando um Resolvedor SAT

Segundo Semestre de 2011

Data de entrega no paca: 30/11/2011

Nota: Iremos usar o resolvedor SAT *MiniSAT*, cujo código pode ser baixado a partir de `http://minisat.se`. O formato de entrada comum aos resolvedores SAT está disponível em `http://www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/satformat.ps`. As fórmulas necessitam estar na forma *clausal*, também conhecida como *forma normal conjuntiva*.

1 Introdução

Para verificar se um número n de exatamente m bits é primo, tentamos encontrar dois números a e b de no máximo $m-1$ bits tal que $n = a \cdot b$. O fato de usarmos $m-1$ bits nos garante[1] que não possamos gerar $n = 1 \cdot n$. Desta forma, se gerarmos uma fórmula que expressa $n = a \cdot b$, e esta fórmula for insatisfatível, então o número n é primo. Se a fórmula é satisfatível, a valoração satisfazedora irá revelar os bits de a e b; itere este processo para produzir todos os fatores primos.

O problema então se reduz a como gerar uma fórmula A tal que, dado n, A é satisfatível se e somente se existem a e b tal que $n = a \cdot b$. A fórmula A será uma (longa) conjunção de cláusulas (uma *cláusula* é a disjunção de literais, e um *literal* é um átomo ou a negação de um átomo) que irão sendo geradas ao longo do processo de codificação.

2 Codificação da Fórmula

Vamos mostrar com um exemplo como podemos codificar a múltiplicação de dois números a e b. Para este exemplo, vamos supor que ambos possuem 3 bits. A multiplicação em binário é muito semelhante a multiplicação usando algarismos decimais:

[1] Porque, como o bit mais significativo de n é 1, teremos $a \neq n$ e $b \neq n$.

$$
\begin{array}{cccccc}
 & & & a_3 & a_2 & a_1 \\
 & & & b_3 & b_2 & b_1 \\
\cline{4-6}
 & & & a_3 \wedge b_1 & a_2 \wedge b_1 & a_1 \wedge b_1 \\
 & & a_3 \wedge b_2 & a_2 \wedge b_2 & a_1 \wedge b_2 & \\
 & a_3 \wedge b_3 & a_2 \wedge b_3 & a_1 \wedge b_3 & & \\
\hline
n_6 & n_5 & n_4 & n_3 & n_2 & n_1
\end{array}
$$

O primeiro passo na codificação será eliminarmos os bits da forma $a_i \wedge b_j$, inserindo novos bits da forma c_{ij}, com o intuito de que $c_{ij} \leftrightarrow a_i \wedge b_j$. Se a e b possuem m bits, estaremos gerando m^2 novos átomos da forma c_{ij}. Cada um destes novos átomos irá contribui com 3 cláusulas para a fórmula A, contendo sua definição $c_{ij} \leftrightarrow a_i \wedge b_j$ no formato clausal:

$$
\begin{array}{l}
\neg c_{ij} \vee a_i \\
\neg c_{ij} \vee b_j \\
c_{ij} \vee \neg a_i \vee \neg b_j
\end{array}
$$

e com isso geramos $3m^2$ cláusulas.

Com isso, a multiplicação fica

$$
\begin{array}{cccccc}
 & & & a_3 & a_2 & a_1 \\
 & & & b_3 & b_2 & b_1 \\
\cline{4-6}
 & & & c_{31} & c_{21} & c_{11} \\
 & & c_{32} & c_{22} & c_{12} & \\
 & c_{33} & c_{23} & c_{13} & & \\
\hline
n_6 & n_5 & n_4 & n_3 & n_2 & n_1
\end{array}
$$

O problema agora se resume a como somar sequências de bits. Antes de detalharmos esta parte, é importante notar que toda vez que um bit x estiver ausentena figura, isto que dizer que $x = 0$. Neste caso iremos propagar este valor zero, da seguinte maneira. Se x é definido como $x \leftrightarrow B$, iremos gerar um conjunto de cláusulas equivalentes a $\neg B$. Por outro lado, se formos somar um bit com zero, a resposta será igual a entrada, portanto, no exemplo acima, já sabemos que $n_1 = c_{11}$.

Para somar duas sequências de bits $c_m \ldots c_1$ com $d_m \ldots d_1$, $2m$ novas variáveis são introduzidas, $e_i, v_i, 1 \leq i \leq m$, onde e é o resultado da soma e v é a sequência de bits "vai-um" auxiliares, de forma que temos

$$
\begin{array}{cccccc}
v_m & v_{m-1} & v_{m-2} & \ldots & v_1 & \\
 & c_m & c_{m-1} & \ldots & c_2 & c_1 \\
+ & d_m & d_{m-1} & \ldots & d_2 & d_1 \\
\hline
e_{m+1} & e_m & e_{m-1} & \ldots & e_2 & e_1
\end{array}
$$

A soma é feita da direita para a esquerda, do bit menos significativo para o mais significativo. Note que cada passo i da soma gera o bit e_i e o vai-um v_i, sendo que este é usado como entrada do próximo passo. As fórmulas que definem os bits e_i e v_i são as seguintes.

$$\begin{array}{l} e_i \leftrightarrow c_i \oplus d_i \oplus v_{i-1} \\ v_i \leftrightarrow (c_i \vee d_i) \wedge (c_i \vee v_{i-1}) \wedge (d_i \vee v_{i-1}) \end{array}$$

onde $\oplus$ é o conectivo "OU-exclusivo", $a \oplus b =_{\text{def}} (a \vee b) \wedge (\neg a \vee \neg b)$.

Para cada i, a primeira fórmula dá origem a 8 cláusulas:

$$\begin{array}{r} \neg c_i \vee \neg d_i \vee \neg v_{i-1} \vee \neg e_i \\ c_i \vee \ \ d_i \vee \neg v_{i-1} \vee \neg e_i \\ \neg c_i \vee \ \ d_i \vee \ \ v_{i-1} \vee \neg e_i \\ c_i \vee \neg d_i \vee \ \ v_{i-1} \vee \neg e_i \\ \neg c_i \vee \ \ d_i \vee \neg v_{i-1} \vee \ \ e_i \\ c_i \vee \neg d_i \vee \neg v_{i-1} \vee \ \ e_i \\ \neg c_i \vee \neg d_i \vee \ \ v_{i-1} \vee \ \ e_i \\ c_i \vee \ \ d_i \vee \ \ v_{i-1} \vee \ \ e_i \end{array}$$

e a segunda fórmula dá origem a 6 cláusulas.

$$\begin{array}{l} \neg v_i \vee \ c_i \vee \ d_i \vee \\ \neg v_i \vee \ c_i \vee \qquad \vee \ v_{i-1} \\ \neg v_i \vee \qquad \vee \ d_i \vee \ v_{i-1} \\ \ v_i \vee \neg c_i \vee \neg d_i \vee \\ \ v_i \vee \neg c_i \vee \qquad \vee \neg v_{i-1} \\ \ v_i \vee \qquad \vee \neg d_i \vee \neg v_{i-1} \end{array}$$

Como são $m-1$ somas que precisaremos realizar, cada uma necessitará de $2m$ variáveis extras e $14m$ cláusulas. Portanto serão $2m(m-1)$ variáveis adicionais para realizar a soma, sendo que as variáveis da resposta final estão contidas nestas (por quê? onde?) e uma geração de $14m(2m-1)$ cláusulas.

No final, as variáveis da resposta final da multiplicação deverão ser identificadas com as variáveis da entrada. Essa identificação pode ser feita da seguinte maneira. Se o bit de entrada correspondente ao resultado r_i for 1, basta adicionar uma cláusula unitária r_i; se for 0, basta adicionar a cláusula unitária $\neg r_i$.

Neste momento, temos uma fórmula A em formato clausal, que pode ser submetida ao resolvedor SAT. Se a fórmula o for satisfatível, teremos na valoração das $2m$ variáveis a e b a decomposição em dois fatores. Caso contrário, a entrada é um número primo.

Note que, no caso de satisfatibilidade, é MUITO SIMPLES testar se a resposta obtida está correta, bastando verificar que $n = a \cdot b$.

2.1 Tamanho da Fórmula

Assumindo que a entrada tenha tamanho $m + 1$.

Se computarmos as variáveis usadas na multiplicação ($3m^2$) e na soma ($2m^2 - 2m$), teremos um total de $5m^2 - 2m$ variáveis inseridas.

O número total de cláusulas inserido na multiplicação será de $3m^2$ e nas somas será de $31m^2 - 14m$.

A razão cláusulas/variáveis é — no limite — 6,2.

3 Entrada e Saída

A entrada do programa é simplesmente um número (em formato binário, mas pode ser decimal, se vocês quiserem).

A resposta será uma lista de pares, uma por linha, cada par contendo o fator primo e o seu expoente na decomposição.

Por exemplo se a entrada for 10011 [ou seja, 18 na base decimal], a resposta será

10	1	[representando 2^1]
11	2	[representando 3^2]

e basta verificar que $18 = 2^1 \cdot 3^2$. Neste exemplo, os fatores primos estão em binário e os expoentes em decimal. Você também poderá apresentar sua resposta com fatores no formato decimal, se desejar.

4 Testes e Implementação

Iremos testar o programa gerando 200 números ÍMPARES aleatórios para m bits, com $m = 15, 20, 25, 30, 35, 40, 45, 50$ bits. Parar antes se o tempo de decomposição médio de um número estiver acima de $100s$. Se estiver muito abaixo dos $100s$, pode continuar para além deste limite. O limite depende da qualidade da codificação, da implementação e da máquina utilizada para os testes, que deverá ser sempre a mesma. Para cada valor de m, registrar o tempo[2] médio de decomposição e o número de fatores primos (sem os expoentes).

O tempo total dos testes pode ser de várias horas.

[2]note que o que importa é o tempo efetivamente rodando, conhecido como *user time*, não o tempo real de execução, que leva em consideração apenas o tempo de início e fim do processo

Note que para gerar aleatoriamente um número ímpar de m bits, tanto o bit mais significativo quanto o menos significativo são 1, portanto basta realizar $m - 2$ sorteios aleatórios.

Implementar o fatorador em uma destas linguagens: C, C++, Java, Pearl, Python ou Prolog. Seu programa deverá gerar arquivos temporários com a fórmula a ser submetida ao resolvedor SAT, invocar o resolvedor e fazer a interpretação da saída para identificar se a fórmula e satisfatível ou não e, em caso positivo, descobrir a valoração das variáveis relativas ao números a e b que compõem os fatores da multiplicação.

Implementar o gerador dos testes, que mede o tempo médio de execução e o número médio de fatores aleatórios

4.1 Entregar

Um único arquivo zipado, contendo:

1. O código do fatorador

2. O código do testador

3. Um pequeno relatório (até 5 páginas em formato pdf) com tabelas e gráficos de Tempo médio $\times$ número de bits (m); e número de fatores primos $\times$ número de bits (m). O seu relatório deverá conter uma breve explicação de como compilar e rodar o programa, e uma discussão dos gráficos apresentados. Mencionar a bibliografia consultada.

Boa sorte e mãos à obra.