Seminar “Formal Grammars”

Boolean Matrix Multiplication (BMM) and CFG parsing

by

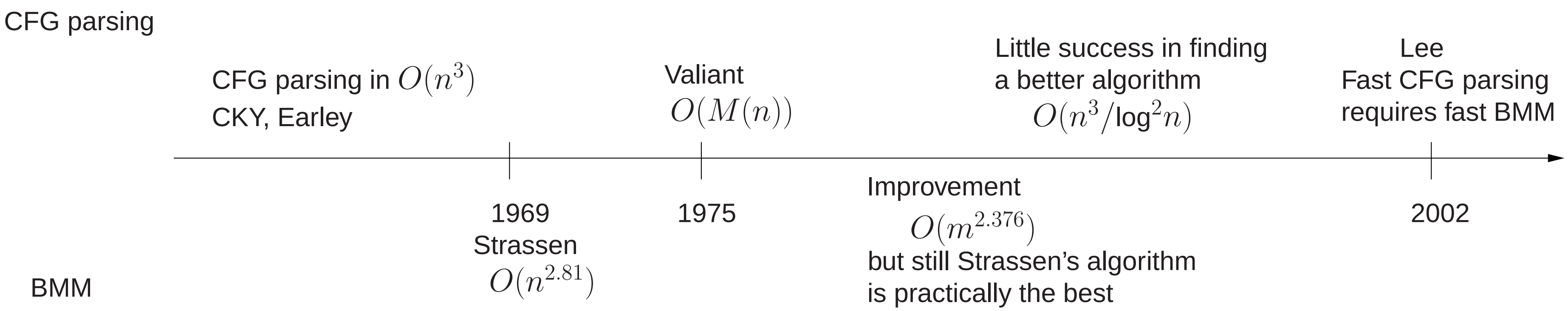
Franziska Ebert

23.3.2007

Overview

1. Motivation
2. Reduction of CFG parsing to BMM (Valiant - 1975)
3. Reverse direction: Conversion of a CFG parser into an algorithm for BMM (Lee - 2002)
4. Conclusion

Motivation


CFG parsing
CFG parsing in $O(n^3)$
CKY, Earley
Valiant
$O(M(n))$
Little success in finding
a better algorithm
$O(n^3/\log^2 n)$
Lee
Fast CFG parsing
requires fast BMM
1969
Strassen
$O(n^{2.81})$
1975
Improvement
$O(m^{2.376})$
but still Strassen's algorithm
is practically the best
2002
BMM

Motivation

CFG parsing

CFG parsing in $O(n^3)$
CKY, Earley

Valiant
$O(M(n))$

Little success in finding
a better algorithm
$O(n^3/\log^2 n)$

Lee
Fast CFG parsing
requires fast BMM

1969
Strassen
$O(n^{2.81})$

1975

Improvement
$O(m^{2.376})$
but still Strassen's algorithm
is practically the best

2002

BMM

Motivation

- General context-free recognition in less than cubic time (Valiant - 1975)
  - CKY, Earley: $O(n^3)$

    $\longrightarrow$ CFG parsing in less than cubic time

    $\longrightarrow$ Reduction of CFG parsing to BMM

    $\longrightarrow$ Today: asymptotically fastest known algorithm

Motivation

- Fast CFG parsing requires fast BMM (Lee - 2002)
  - Little success in developing practical fast CFG parsers
  - Practically fastest BMM algorithm: Strassen - $O(n^{2.81})$

    $\longrightarrow$ No practical fast BMM algorithm

    $\longrightarrow$ Proof: CFG parsing relies on BMM

    $\longrightarrow$ Conversion of a CFG parser ($O(gn^{3-\epsilon})$) into an algorithm for BMM ($O(m^{3-\epsilon/3})$)

Reduction of CFG parsing to BMM

- General idea for an MM algorithm to parse CFG

  $\longrightarrow$ Build upper triangular matrix

  $\longrightarrow$ Define new matrix multiplication

  $\longrightarrow$ Matrix has the form of CKY recognition matrix

  $\longrightarrow A_k \in b_{ij} \Leftrightarrow A_k \longrightarrow^* w_i^{j-1}$

$$\begin{pmatrix} \emptyset & \{A\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \{A\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{B\} & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \{B\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix} \Rightarrow^* \begin{pmatrix} \emptyset & \{A\} & \{X\} & \emptyset & \{S\} \\ \emptyset & \emptyset & \{A\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{B\} & \{Y\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \{B\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

Preliminaries

- CFG (N, Σ, P, A_1) in Chomsky normal form:
  - N: set of nonterminals $\{A_1, \ldots, A_h\}$
  - Σ: set of terminals
  - P: set of productions
  - A_1: the starting symbol
- $A_i \to^* w$ denotes: w can be derived from A_i

Preliminaries

Definition: $*$**-operator on subsets of N**

$$N_1 * N_2 = \{A_i \mid \exists A_j \in N_1, A_k \in N_2.(A_i \to A_j A_k) \in P\}$$

Example:

$$X \to YZ \qquad \{A, Y\} * \{Z\} = \{X\}$$

Definition: **Matrix Multiplication over matrices with subsets of N as elements.**

$$a * b = c :\Leftrightarrow c_{ik} = \bigcup_{j=1}^{n} a_{ij} * b_{jk}$$

Preliminaries

Definition: Transitive Closure of a square matrix

$$a^+ = a^{(1)} \cup a^{(2)} \cup \ldots$$

where

$$a^{(i)} = \bigcup_{j=1}^{i-1} a^{(j)} * a^{(i-j)} \qquad \text{and} \qquad a^{(1)} = a$$

Definition: Union of two matrices

$$a \cup b = c :\Leftrightarrow c_{ij} = a_{ij} \cup b_{ij}$$

Reduction of CFG parsing to BMM

- Input: CFG $G = (N, \Sigma, P, A_1)$ and $w = x_1 \ldots x_n$
- Output: $A_1 \rightarrow^* w$?
- Algorithm: Let b be a $(n+1) \times (n+1)$-matrix

1. $b_{i,j} = \emptyset \qquad \forall 1 \leq i, j \leq (n+1)$
2. $b_{i,i+1} = \{A_k \mid (A_k \rightarrow x_i) \in P\}$
3. Compute $a = b^+$
4. Check whether $A_1 \in a_{1,n+1}$

Reduction of CFG parsing to BMM - Example

$$S \rightarrow XY$$

$$X \rightarrow XA \mid AA$$

$$Y \rightarrow YB \mid BB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$w = aabb$$

$$a := \begin{pmatrix} \emptyset & \{A\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \{A\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{B\} & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \{B\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

$$b_{i,i+1} = \{A_k \mid (A_k \rightarrow x_i) \in P\}$$

Reduction of CFG parsing to BMM - Example

$$S \rightarrow XY$$
$$X \rightarrow XA \mid AA$$
$$Y \rightarrow YB \mid BB$$
$$A \rightarrow a$$
$$B \rightarrow b$$

$$w = aabb$$

$$a := \begin{pmatrix} \emptyset & \{A\} & \{X\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \{A\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{B\} & \{Y\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \{B\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

$$a := \underbrace{a^{(1)}}_{=a} \cup \underbrace{a^{(2)}}_{=a^{(1)} * a^{(1)}}$$

Reduction of CFG parsing to BMM - Example

$$S \rightarrow XY$$
$$X \rightarrow XA \mid AA$$
$$Y \rightarrow YB \mid BB$$
$$A \rightarrow a$$
$$B \rightarrow b$$

$$w = aabb$$

$$a := \begin{pmatrix} \emptyset & \{A\} & \{X\} & \emptyset & \{S\} \\ \emptyset & \emptyset & \{A\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{B\} & \{Y\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \{B\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

$$a := a^{(1)} \cup a^{(2)} \cup \underbrace{a^{(3)}}_{=a^{(1)} * a^{(2)} \cup a^{(2)} * a^{(1)}}$$

Time Bounds

- Time complexity: $O(n^{2.81})$
- $T(n)$:= time to compute transitive closure
- $BM(n)$:= complexity for BMM algorithm
- $M(n)$:= complexity for multiplying two matrices

$T(n) \leq M(n) \cdot const$ → $M(n) \leq BM(n) \cdot const$ → $O(BM(n))$

Time complexity for CFG parsing: $O(n^{2.81})$

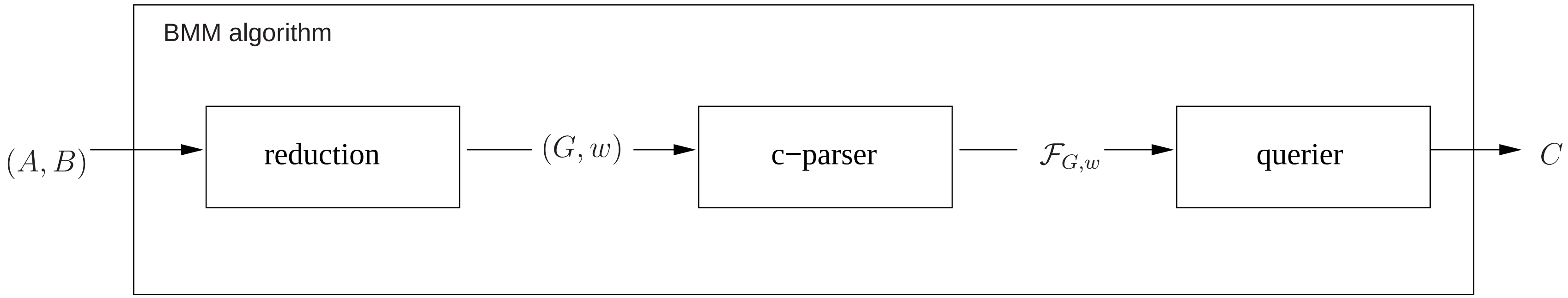
An Algorithm for BMM

Fundamental Theorem:

Fast context-free grammar parsing requires fast boolean matrix multiplication.

$\longrightarrow$ Any CFG-parser running in $O(gn^{3-\epsilon})$ can be converted with little computational overhead into an $O(m^{3-\epsilon/3})$ BMM algorithm

An Algorithm for BMM



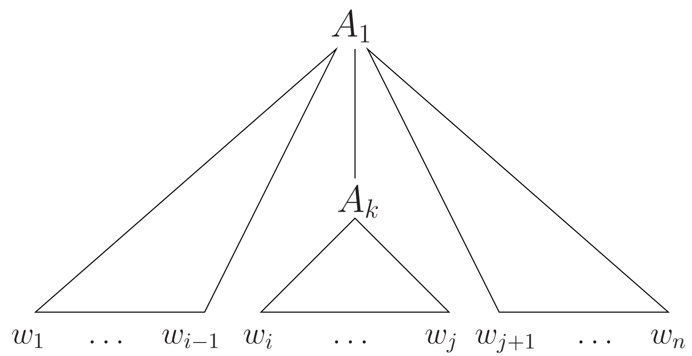
Conversion of a CFG parser into an algorithm for BMM

Preliminaries

Definition: c-derivation

$A_k \in N$ c-derives w_i^j iff

- $A_k \rightarrow^* w_i^j$, and
- $A_1 \rightarrow^* w_1^{i-1} A_k w_{j+1}^n$



Definition: c-parser

A c-parser is an algorithm that takes a CFG G and a string w and outputs $\mathcal{F}_{G,w}$ with:

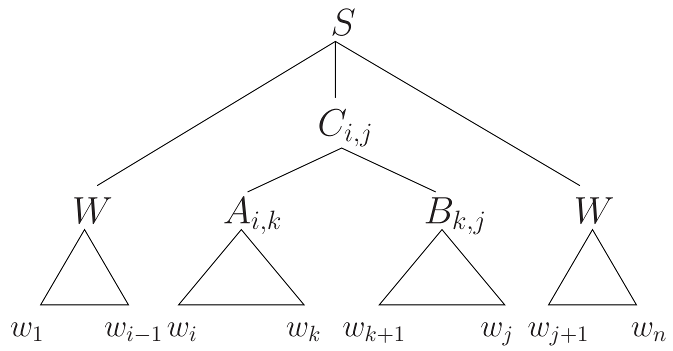
- A_k c-derives $w_i^j \Rightarrow \mathcal{F}_{G,w} =$ "yes"
- $A_k \not\rightarrow^* w_i^j \Rightarrow \mathcal{F}_{G,w} =$ "no"
- $\mathcal{F}_{G,w}$ answers queries in constant time.

Reduction: $(A, B) \to (G, w)$

- A,B are $m \times m$ boolean matrices
- All information about A and B is coded in the grammar
  $\to$ string w does not depend on A and B.
- Recall: $C = A * B$ is defined as $c_{ij} = \bigvee_{k=1}^{m}(a_{ik} \wedge b_{kj})$
  $\to c_{ij} = 1 \leftrightarrow \exists k. a_{ik} = b_{kj} = 1$

Reduction: $(A, B) \to (G, w)$

- General Idea:
  - For each $a_{ij} = 1$ introduce production $A_{i,j} \to w_i W w_j$
  - For each $b_{ij} = 1$ introduce production $B_{i,j} \to w_{i+1} W w_j$
  - For each c_{ij} introduce production $C_{i,j} \to A_{i,k} B_{k,j}\ \forall 1 \le k \le m$
  - $c_{ij} = 1$ iff $C_{i,j}$ c-derives w_i^j



Reduction: $(A, B) \longrightarrow (G, w)$

- To obtain smaller complexity:

  $\longrightarrow$ Keep grammar size small

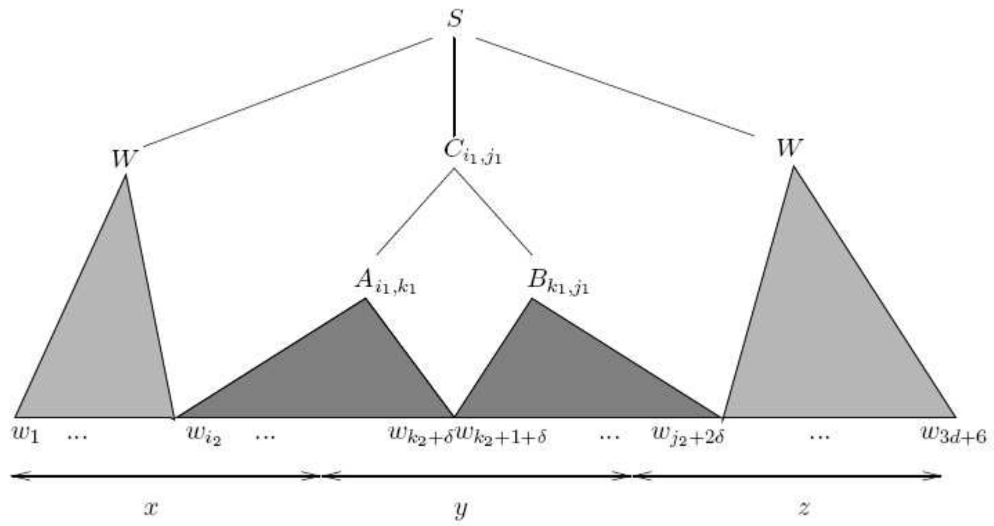
  $\longrightarrow$ Split indices, e.g. $i = (i_1, i_2)$

- $i_1 = \lfloor i/d \rfloor$ and $i_2 = (i \text{ mod } d) + 2$, with $d = \lceil m^{1/3} \rceil$

  $\longrightarrow 0 \leq i_1 \leq d^2$ and $2 \leq i_2 \leq d + 1$

  $\longrightarrow i$ can be computed uniquely by i_1 and i_2

Reduction: $(A, B) \rightarrow (G, w)$



- $A_{i_1,j_1} \rightarrow w_{i_2} W w_{j_2+\delta}$, with $\delta := d + 2$
- String $w = w_1 \ldots w_\delta w_{\delta+1} \ldots w_{2\delta} w_{2\delta+1} \ldots w_{3\delta}$
  $\rightarrow \Sigma = \{w_l \mid 1 \leq l \leq 3\delta = 3d + 6\}$

Reduction: $(A, B) \rightarrow (G, w)$

- W-rules: $W \rightarrow w_l W \mid w_l$
- A-rules: $a_{ij} = 1 \Rightarrow A_{i_1,j_1} \rightarrow w_{i_2} W w_{j_2+\delta}$
- B-rules: $b_{ij} = 1 \Rightarrow B_{i_1,j_1} \rightarrow w_{i_2+1+\delta} W w_{j_2+2\delta}$
- C-rules: $C_{i_1,j_1} \rightarrow A_{i_1,k_1} B_{k_1,j_1}$
- S-rules: $S \rightarrow W C_{i_1,j_1} W$

Reduction: $(A, B) \to (G, w)$

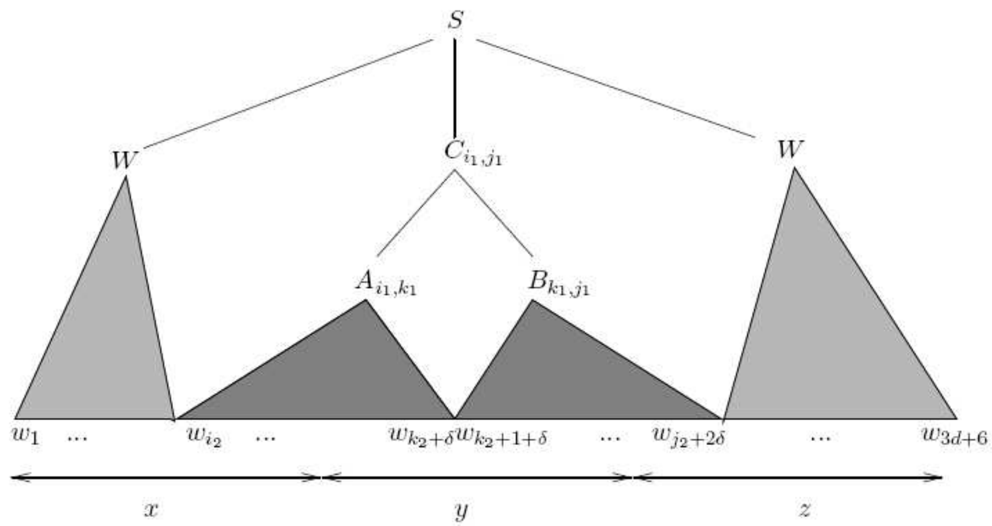
Theorem:

For $1 \le i, j \le m$, the entry c_{ij} is 1 iff C_{i_1,j_1} c-derives $w_{i_2}^{j_2+2\delta}$.

Proof: "$\Rightarrow$"

to show: C_{i_1,j_1} c-derives $w_{i_2}^{j_2+2\delta}$

$$
\begin{aligned}
C_{i_1,j_1} &\to A_{i_1,k_1} B_{k_1,j_1} && \text{C-rule} \\
&\to^* w_{i_2} W w_{k_2+\delta}\, w_{k_2+\delta+1} W w_{j_2+2\delta} && \exists k . a_{ik} = b_{kj} = 1 \\
&\to^* w_{i_2}^{j_2+2\delta} && \text{W-rule}
\end{aligned}
$$



- $C_{i_1,j_1} \longrightarrow^* w_{i_2}^{j_2+2\delta}$
- S-rule: $S \longrightarrow W C_{i_1,j_1} W$

$\Rightarrow$ C_{i_1,j_1} c-derives $w_{i_2}^{j_2+2\delta}$

Reduction: $(A, B) \to (G, w)$

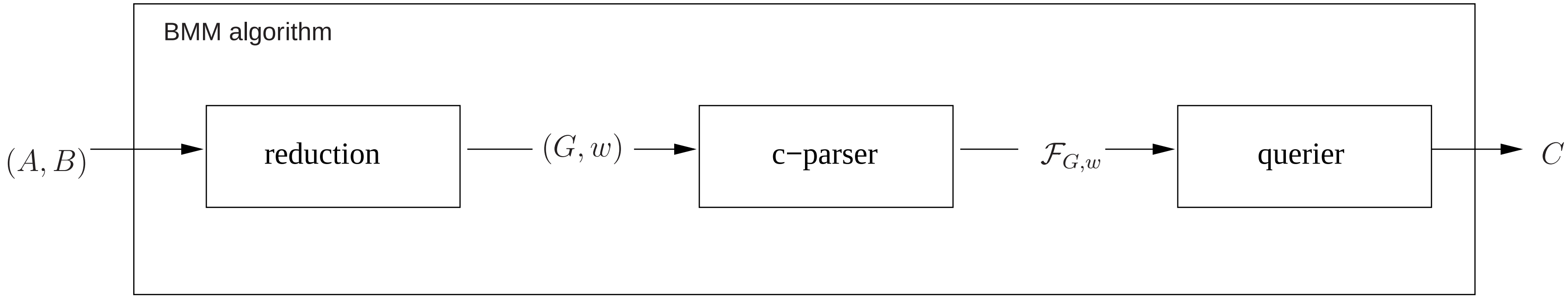
Theorem:
For $1 \leq i, j \leq m$, the entry c_{ij} is 1 iff C_{i_1,j_1} c-derives $w_{i_2}^{j_2+2\delta}$.

Proof: "$\Leftarrow$"

to show: $c_{ij} = 1$

$$
\begin{aligned}
& C_{i_1,j_1} \to^* w_{i_2}^{j_2+2\delta} \\
& \Rightarrow \exists k_1 . C_{i_1,j_1} \to A_{i_1,k_1} B_{k_1,j_1} \\
& \quad \to w_{i_2} W w_{k_2+\delta} w_{k_2+\delta+1} W w_{j_2+2\delta} \to^* w_{i_2}^{j_2+2\delta} \\
& \Rightarrow \exists k = (k_1, k_2) . a_{ik} = b_{kj} = 1 \\
& \Rightarrow c_{ij} = 1
\end{aligned}
$$

An Algorithm for BMM



- Querier: for all c_{ij} ask $\mathcal{F}_{G,w}$ whether C_{i_1,j_1} c-derives $w_{i_2}^{j_2+2\delta}$

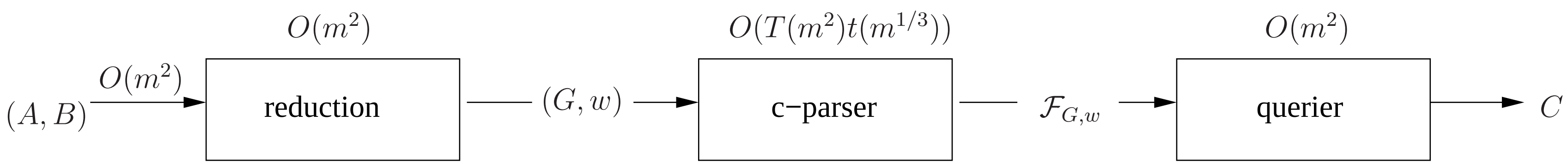
Conclusion: For two boolean matrices A and B we can compute A*B with a c-parser.

Time bounds

Theorem:

Any c-parser P with running time $O(T(g)t(n))$ can be converted into a BMM algorithm M_P that runs in time $O(max(m^2, T(m^2)t(m^{1/3})))$. In particular, if P takes time $O(gn^{3-\epsilon})$, then M_P runs in time $O(m^{3-\epsilon/3})$.

Proof:



- with $|G| = O(m^2)$, $|w| = O(m^{1/3})$
- if $T(g) = g$ and $t(n) = n^{3-\epsilon}$
  $\rightarrow O(m^2(m^{1/3})^{3-\epsilon}) = O(m^{3-\epsilon/3})$

Conclusion

- Reduction: CFG parsing to BMM
- Reverse direction: Using a CFG parser for BMM
- No faster CFG parsing without finding a faster BMM algorithm
- No faster BMM without finding a faster CFG parsing algorithm
- Related Work: Improvement of TAG parsing yields sub-cubic BMM (Satta, 1994)

References

- Leslie G. Valiant, 1975. General context-free recognition in less than cubic time. Journal of Computer Science, 10:308-315
- Lillian Lee, 2002. Fast context-free grammar parsing requires fast boolean matrix multiplication. Journal of the ACM 49(1):1-15
- Giorgio Satta, 1994. Tree-adjoining grammar parsing and Boolean matrix multiplication. Computational Linguistics, 20(2):173-191