Compression of Rewriting Systems for Termination Analysis

A. Bau, M. Lohrey, E. Noeth, J. Waldmann

RTA'13, Eindhoven

Example (String Rewriting)

- Given a linear interpretation $[\cdot]$ for $\Sigma = \{a, b\}$, how would you compute $[aabb], [bbbaaa]$?
- Naive computation: $[a[a[bb]]], [b[b[b[a[aa]]]]]$ total 8 multiplications
- More efficient: $c = [aa], d = [bb], [cd], [b[d[ca]]]$ total 6 multiplications
- Concrete $\to$ symbolic computation (produce a constraint system that describes compatibility of unknown interpretation with rewrite system)
- Goal: Compress terms. Questions: W.r.t. which cost measure? Efficient compression algorithm (perhaps approximative)

Example (String Rewriting)

- Given a linear interpretation $[\cdot]$ for $\Sigma = \{a, b\}$, how would you compute $[aabb], [bbbaaa]$?
- Naive computation: $[a[a[bb]]], [b[b[b[a[aa]]]]]$ total 8 multiplications
- More efficient: $c = [aa], d = [bb], [cd], [b[d[ca]]]$ total 6 multiplications
- Concrete $\rightarrow$ symbolic computation (produce a constraint system that describes compatibility of unknown interpretation with rewrite system)
- Goal: Compress terms. Questions: W.r.t. which cost measure? Efficient compression algorithm (perhaps approximative)

Example (String Rewriting)

- Given a linear interpretation $[\cdot]$ for $\Sigma = \{a, b\}$, how would you compute $[aabb], [bbbaaa]$?
- Naive computation: $[a[a[bb]]], [b[b[b[a[aa]]]]]$ total 8 multiplications
- More efficient: $c = [aa], d = [bb], [cd], [b[d[ca]]]$ total 6 multiplications
- Concrete $\to$ symbolic computation (produce a constraint system that describes compatibility of unknown interpretation with rewrite system)
- Goal: Compress terms. Questions: W.r.t. which cost measure? Efficient compression algorithm (perhaps approximative)

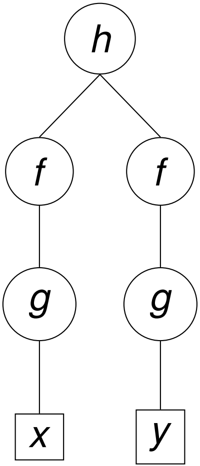
Example (String Rewriting)

- Given a linear interpretation $[\cdot]$ for $\Sigma = \{a, b\}$, how would you compute $[aabb], [bbbaaa]$?
- Naive computation: $[a[a[bb]]], [b[b[b[a[aa]]]]]$ total 8 multiplications
- More efficient: $c = [aa], d = [bb], [cd], [b[d[ca]]]$ total 6 multiplications
- Concrete $\rightarrow$ symbolic computation (produce a constraint system that describes compatibility of unknown interpretation with rewrite system)
- Goal: Compress terms. Questions: W.r.t. which cost measure? Efficient compression algorithm (perhaps approximative)

Example (String Rewriting)

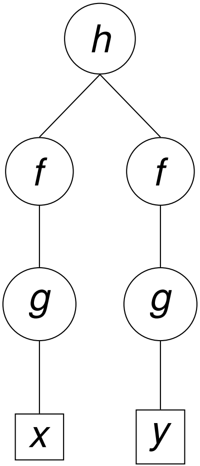
- Given a linear interpretation $[\cdot]$ for $\Sigma = \{a, b\}$, how would you compute $[aabb], [bbbaaa]$?
- Naive computation: $[a[a[bb]]], [b[b[b[a[aa]]]]]$ total 8 multiplications
- More efficient: $c = [aa], d = [bb], [cd], [b[d[ca]]]$ total 6 multiplications
- Concrete $\rightarrow$ symbolic computation (produce a constraint system that describes compatibility of unknown interpretation with rewrite system)
- Goal: Compress terms. Questions: W.r.t. which cost measure? Efficient compression algorithm (perhaps approximative)

Example (Term Rewriting)



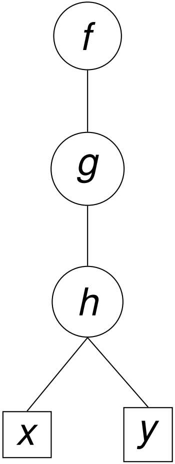
- Normal bottom-up evaluation: Four multiplications
- Instead, evaluate $F_1 \cdot G_1$ first: Three multiplications (we will call $(f, 1, g)$ a *digram with positive sharing)*

Example (Term Rewriting)



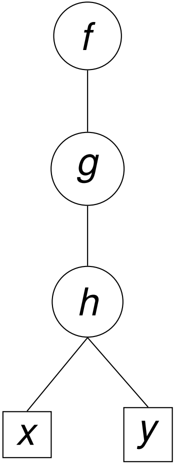
- Normal bottom-up evaluation: Four multiplications
- Instead, evaluate $F_1 \cdot G_1$ first: Three multiplications (we will call $(f, 1, g)$ a *digram with positive sharing)*

Example (Term Rewriting)



- Normal bottom-up evaluation: Four multiplications: $F_1 \cdot G_1 \cdot H_1$ and $F_1 \cdot G_1 \cdot H_2$,
- Instead, evaluate $F_1 \cdot G_1$ first.

Example (Term Rewriting)



- Normal bottom-up evaluation: Four multiplications: $F_1 \cdot G_1 \cdot H_1$ and $F_1 \cdot G_1 \cdot H_2$,
- Instead, evaluate $F_1 \cdot G_1$ first.

Overview

- Introduction
- Cost Function
- Compression by Digrams
- Adaption of TreeRePair
- Experiments

Cost of Terms

- Consider matrix interpretations: k-ary symbol f interpreted by function

$$(x_1, \ldots, x_k) \mapsto F_0 + F_1 x_1 + \ldots + F_k x_k$$

  where F_0 vector, $F_1, \ldots, F_k$ matrices
- Interpretation of term $t \in \mathrm{Term}(\Sigma, V)$ is function $[t]$ also of this shape with $|\mathrm{Var}(t)|$ arguments
- Cost of a term t is number of matrix-by-matrix multiplications needed to compute $[t]$ bottom-up

Cost of Terms

- Consider matrix interpretations: k-ary symbol f interpreted by function

$$(x_1, \ldots, x_k) \mapsto F_0 + F_1 x_1 + \ldots + F_k x_k$$

where F_0 vector, $F_1, \ldots, F_k$ matrices

- Interpretation of term $t \in \mathrm{Term}(\Sigma, V)$ is function $[t]$ also of this shape with $|\mathrm{Var}(t)|$ arguments
- Cost of a term t is number of matrix-by-matrix multiplications needed to compute $[t]$ bottom-up

Cost of Terms

- Consider matrix interpretations: k-ary symbol f interpreted by function

$$(x_1, \ldots, x_k) \mapsto F_0 + F_1 x_1 + \ldots + F_k x_k$$

  where F_0 vector, $F_1, \ldots, F_k$ matrices
- Interpretation of term $t \in \mathrm{Term}(\Sigma, V)$ is function $[t]$ also of this shape with $|\mathrm{Var}(t)|$ arguments
- Cost of a term t is number of matrix-by-matrix multiplications needed to compute $[t]$ bottom-up

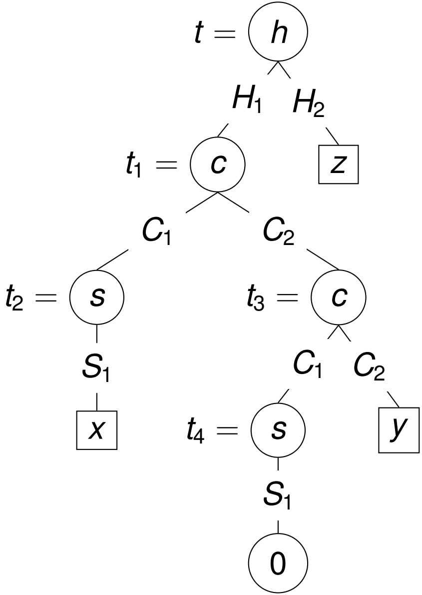
Cost of Terms

Definition

The *(matrix multiplication) cost* of a term $t = (D, \lambda) \in \mathrm{Term}(\Sigma, V)$ is

$$\mathrm{cost}(t) = \sum_{p \in D \setminus \{\varepsilon\}, \lambda(p) \notin V} |\,\mathrm{Var}(t|_p)|.$$

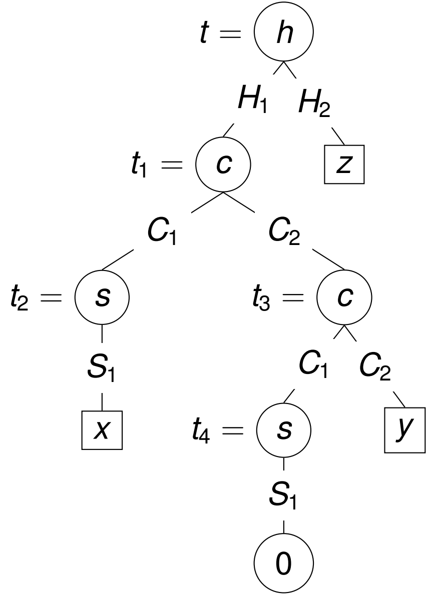
The cost of a tuple $(t_1, \ldots, t_m)$ of terms is $\sum_{i=1}^{m} \mathrm{cost}(t_i)$.



Example:

$$h(c(s(x), c(s(0), y)), z)$$

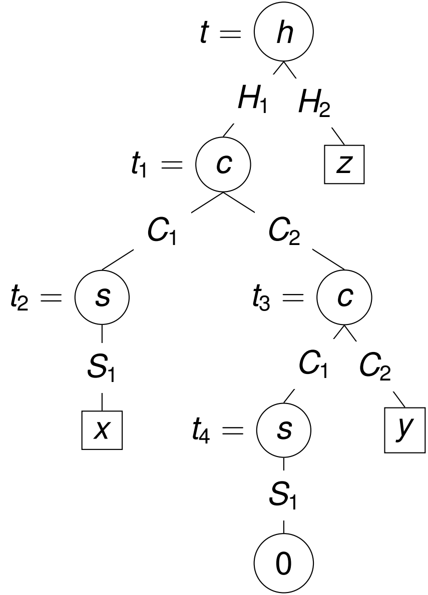
- $\text{cost}(t_4) = 0$
- $\text{cost}(t_2) = 1$
- $\text{cost}(t_3) = 1$
- $\text{cost}(t_1) = 2$
- $\text{cost}(t) = 4$



Example:

$$h(c(s(x), c(s(0), y)), z)$$

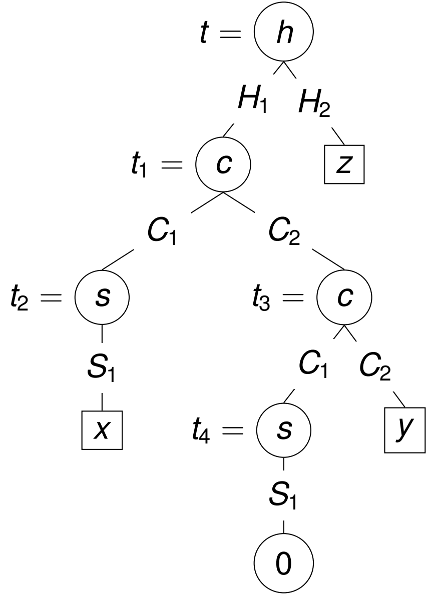
- $\mathrm{cost}(t_4) = 0$
- $\mathrm{cost}(t_2) = 1$
- $\mathrm{cost}(t_3) = 1$
- $\mathrm{cost}(t_1) = 2$
- $\mathrm{cost}(t) = 4$



Example:

$$h(c(s(x), c(s(0), y)), z)$$

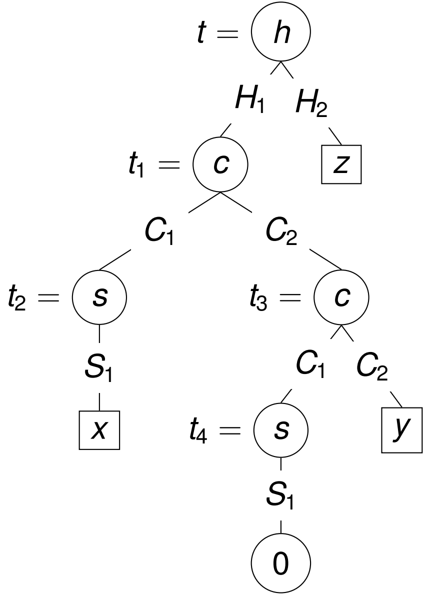
- $\mathrm{cost}(t_4) = 0$
- $\mathrm{cost}(t_2) = 1$
- $\mathrm{cost}(t_3) = 1$
- $\mathrm{cost}(t_1) = 2$
- $\mathrm{cost}(t) = 4$



Example:

$$h(c(s(x), c(s(0), y)), z)$$

- $\mathrm{cost}(t_4) = 0$
- $\mathrm{cost}(t_2) = 1$
- $\mathrm{cost}(t_3) = 1$
- $\mathrm{cost}(t_1) = 2$
- $\mathrm{cost}(t) = 4$



Example:

$$h(c(s(x), c(s(0), y)), z)$$

- $\text{cost}(t_4) = 0$
- $\text{cost}(t_2) = 1$
- $\text{cost}(t_3) = 1$
- $\text{cost}(t_1) = 2$
- $\text{cost}(t) = 4$

Compression by Digrams

- A *digram* is $h = (f, i, g)$ where $f \in \Sigma_k, g \in \Sigma_l$.
  This is a $(k - 1 + l)$-ary symbol, with *expansion*
  $h(x_1, \dots, x_{i-1}, y_1, \dots, y_l, x_{i+1}, \dots, x_k) \to$
  $f(x_1, \dots, x_{i-1}, g(y_1, \dots, y_l), x_{i+1}, \dots, x_k)$
- The cost of a digram $h = (f, i, g)$ is the arity of g because that many multiplications are needed to get the coefficients of $(y_1, \dots, y_l)$
- The set $S \subseteq \text{Term}(\Sigma, V)$ can be represented by set $S' \subseteq \text{Term}(\Sigma \cup D, V)$,
  where $D =$ digrams (possibly nested),
  $S = \text{expand}(S')$
- Cost of S' is cost of digrams $+$ cost of terms

Compression by Digrams

- A *digram* is $h = (f, i, g)$ where $f \in \Sigma_k, g \in \Sigma_l$. This is a $(k - 1 + l)$-ary symbol, with *expansion* $h(x_1, \dots, x_{i-1}, y_1, \dots, y_l, x_{i+1}, \dots, x_k) \to f(x_1, \dots, x_{i-1}, g(y_1, \dots, y_l), x_{i+1}, \dots, x_k)$
- The cost of a digram $h = (f, i, g)$ is the arity of g because that many multiplications are needed to get the coefficients of $(y_1, \dots, y_l)$
- The set $S \subseteq \mathrm{Term}(\Sigma, V)$ can be represented by set $S' \subseteq \mathrm{Term}(\Sigma \cup D, V)$, where $D =$ digrams (possibly nested), $S = \mathrm{expand}(S')$
- Cost of S' is cost of digrams $+$ cost of terms

Compression by Digrams

- A *digram* is $h = (f, i, g)$ where $f \in \Sigma_k, g \in \Sigma_l$. This is a $(k - 1 + l)$-ary symbol, with *expansion* $h(x_1, \dots, x_{i-1}, y_1, \dots, y_l, x_{i+1}, \dots, x_k) \to f(x_1, \dots, x_{i-1}, g(y_1, \dots, y_l), x_{i+1}, \dots, x_k)$
- The cost of a digram $h = (f, i, g)$ is the arity of g because that many multiplications are needed to get the coefficients of $(y_1, \dots, y_l)$
- The set $S \subseteq \mathrm{Term}(\Sigma, V)$ can be represented by set $S' \subseteq \mathrm{Term}(\Sigma \cup D, V)$,
  where $D =$ digrams (possibly nested),
  $S = \mathrm{expand}(S')$
- Cost of S' is cost of digrams $+$ cost of terms

Compression by Digrams

- A *digram* is $h = (f, i, g)$ where $f \in \Sigma_k, g \in \Sigma_l$. This is a $(k - 1 + l)$-ary symbol, with *expansion* $h(x_1, \ldots, x_{i-1}, y_1, \ldots, y_l, x_{i+1}, \ldots, x_k) \to f(x_1, \ldots, x_{i-1}, g(y_1, \ldots, y_l), x_{i+1}, \ldots, x_k)$
- The cost of a digram $h = (f, i, g)$ is the arity of g because that many multiplications are needed to get the coefficients of $(y_1, \ldots, y_l)$
- The set $S \subseteq \mathrm{Term}(\Sigma, V)$ can be represented by set $S' \subseteq \mathrm{Term}(\Sigma \cup D, V)$,
  where $D =$ digrams (possibly nested),
  $S = \mathrm{expand}(S')$
- Cost of S' is cost of digrams $+$ cost of terms

Example (Digrams)

Example

Compressed term list:

$$([h,2,c](x,y,z),\ [[h,1,c],1,s](y,x,z) \mid$$
$$[h,1,c],\ [h,2,c],[[h,1,c],1,s]$$

Expansion:

$$(h(x,c(y,z)),\ h(c(s(y),x),z)$$

Example continued

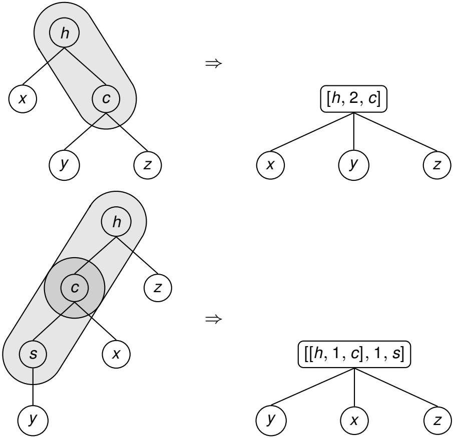


Figure : The replaced digrams from the previous example.

Previous work (Lohrey et al. (2011))

- Unit cost (each node costs 1) models size compression,
- Has been used to compress XML documents
- The exact compression problem is NP-hard
- Approximative, iterative algorithm: in each step pick the digram with largest *savings*
- On-the-fly update of savings is possible (results in linear time algorithm)

Previous work (Lohrey et al. (2011))

- Unit cost (each node costs 1) models size compression,
- Has been used to compress XML documents
- The exact compression problem is NP-hard
- Approximative, iterative algorithm: in each step pick the digram with largest *savings*
- On-the-fly update of savings is possible (results in linear time algorithm)

Previous work (Lohrey et al. (2011))

- Unit cost (each node costs 1) models size compression,
- Has been used to compress XML documents
- The exact compression problem is NP-hard
- Approximative, iterative algorithm: in each step pick the digram with largest *savings*
- On-the-fly update of savings is possible (results in linear time algorithm)

Previous work (Lohrey et al. (2011))

- Unit cost (each node costs 1) models size compression,
- Has been used to compress XML documents
- The exact compression problem is NP-hard
- Approximative, iterative algorithm: in each step pick the digram with largest *savings*
- On-the-fly update of savings is possible (results in linear time algorithm)

Previous work (Lohrey et al. (2011))

- Unit cost (each node costs 1) models size compression,
- Has been used to compress XML documents
- The exact compression problem is NP-hard
- Approximative, iterative algorithm: in each step pick the digram with largest *savings*
- On-the-fly update of savings is possible (results in linear time algorithm)

TreeRePair (Lohrey et al 2011)

input: a term list $\bar{t} = (t_1, \ldots, t_m)$
$\bar{d} := \varepsilon$ (a list of digrams)
while there exists a digram d with $\text{maxSize}(d, \bar{t}) > 1$
do

let d be a digram with
$\text{maxSize}(d, \bar{t}) \geq \text{maxSize}(d', \bar{t})$ for all digrams d'
let $\bar{u}$ such that $\bar{t} \to_{\text{maxOcc}(d,\bar{t})} \bar{u}$
$\bar{t} := \bar{u}$; $\bar{d} := (\bar{d}, d)$

endwhile
output: $(\bar{t} \mid \bar{d})$

$\text{maxOcc}(d, \bar{t})$: max. list of non-overlapping digrams
$\text{maxSize}(d, \bar{t}) := |\text{maxOcc}(d, \bar{t})|$

Our contributions

- Define non-uniform cost model, suitable for computing coefficients of linear interpretations
- Implement efficiently (keep the algorithmic idea of TreeRePair)
- Adapt to the dependency pairs transformation

Our contributions

- Define non-uniform cost model, suitable for computing coefficients of linear interpretations
- Implement efficiently (keep the algorithmic idea of TreeRePair)
- Adapt to the dependency pairs transformation

Our contributions

- Define non-uniform cost model, suitable for computing coefficients of linear interpretations
- Implement efficiently (keep the algorithmic idea of TreeRePair)
- Adapt to the dependency pairs transformation

Compression and DP Transform

- Dependency Pairs transformation creates (many) additional rules, in extended (marked) signature
- matrix interpretations for DP transformed systems use two-sorted algebra (base sort: vectors, top sort: scalars)
- interpretation of marked terms can be done in the top sort completely (starting from the top, vector-by-matrix multiplications only, not matrix-by-matrix)
- compress the original system as usual

Compression and DP Transform

- Dependency Pairs transformation creates (many) additional rules, in extended (marked) signature
- matrix interpretations for DP transformed systems use two-sorted algebra (base sort: vectors, top sort: scalars)
- interpretation of marked terms can be done in the top sort completely (starting from the top, vector-by-matrix multiplications only, not matrix-by-matrix)
- compress the original system as usual

Compression and DP Transform

- Dependency Pairs transformation creates (many) additional rules, in extended (marked) signature
- matrix interpretations for DP transformed systems use two-sorted algebra (base sort: vectors, top sort: scalars)
- interpretation of marked terms can be done in the top sort completely (starting from the top, vector-by-matrix multiplications only, not matrix-by-matrix)
- compress the original system as usual

Compression and DP Transform

- Dependency Pairs transformation creates (many) additional rules, in extended (marked) signature
- matrix interpretations for DP transformed systems use two-sorted algebra (base sort: vectors, top sort: scalars)
- interpretation of marked terms can be done in the top sort completely (starting from the top, vector-by-matrix multiplications only, not matrix-by-matrix)
- compress the original system as usual

Experiments-Settings

- We use restricted version of Matchbox (to isolate the compression effect)
  `https://github.com/jwaldmann/matchbox`
- Matrix interpretations as only non-cheap method
- Four settings: No compression, compression, Dependency Pairs w/o compression and DP with compression

Experiments-Settings

- We use restricted version of Matchbox (to isolate the compression effect)

  `https://github.com/jwaldmann/matchbox`
- Matrix interpretations as only non-cheap method
- Four settings: No compression, compression, Dependency Pairs w/o compression and DP with compression

Experiments-Settings

- We use restricted version of Matchbox (to isolate the compression effect)
  `https://github.com/jwaldmann/matchbox`
- Matrix interpretations as only non-cheap method
- Four settings: No compression, compression, Dependency Pairs w/o compression and DP with compression

Performance Data (from TPDB)

method	cost	CNF-size (var, cl.)
no compression	$1.61 \cdot 10^6$	$4.04 \cdot 10^8, 3.23 \cdot 10^9$
compression	$5.18 \cdot 10^5$	$1.30 \cdot 10^8, 1.04 \cdot 10^9$
dependency pairs (DP)	$1.51 \cdot 10^6$	$1.92 \cdot 10^9, 6.22 \cdot 10^9$
DP and compression	$4.39 \cdot 10^5$	$1.11 \cdot 10^9, 3.63 \cdot 10^9$

Table : Total cost and CNF-size with and without compression, for 3027 systems from TPDB

Both costs and CNF-size are approximately $1/3$ of the size of their non- compressed counterparts.

Performance Data (from TPDB)

method	av. time yes	# yes inst.
no compression	11.9	584
compression with MCTreeRePair	12.2	628
dependency pairs (DP)	1.85	681
DP and compression	4.10	709

Table : Influence of compression on the matchbox termination prover.

Discussion

- Cost models matrix-by-matrix multiplications only
- Matrix-by-vector (for absolute coefficients)?
- Vector-by-matrix (for marked terms)?
- …and what about additions?
- In general, matrix multiplication chain problem

Discussion

- Cost models matrix-by-matrix multiplications only
- Matrix-by-vector (for absolute coefficients)?
- Vector-by-matrix (for marked terms)?
- . . . and what about additions?
- In general, matrix multiplication chain problem

Discussion

- Cost models matrix-by-matrix multiplications only
- Matrix-by-vector (for absolute coefficients)?
- Vector-by-matrix (for marked terms)?
- …and what about additions?
- In general, matrix multiplication chain problem

Discussion

- Cost models matrix-by-matrix multiplications only
- Matrix-by-vector (for absolute coefficients)?
- Vector-by-matrix (for marked terms)?
- ... and what about additions?
- In general, matrix multiplication chain problem

Discussion

- Cost models matrix-by-matrix multiplications only
- Matrix-by-vector (for absolute coefficients)?
- Vector-by-matrix (for marked terms)?
- ... and what about additions?
- In general, matrix multiplication chain problem