Linnea: Automatic Generation of Efficient Linear Algebra Programs

Henrik Barthels

HPAC High Performance and Automatic Computing

RWTH AACHEN UNIVERSITY

Introduction

- How to compute the following expressions?

$$b := (X^T X)^{-1} X^T y$$
$$b := (X^T M^{-1} X)^{-1} X^T M^{-1} y$$
$$x := W(A^T (AWA^T)^{-1} b - c)$$
$$x := \left(A^{-T} B^T B A^{-1} + R^T [\Lambda(Rz)] R\right)^{-1} A^{-T} B^T B A^{-1} y$$

Introduction

- How to compute the following expressions?

$$b := (X^T X)^{-1} X^T y$$
$$b := (X^T M^{-1} X)^{-1} X^T M^{-1} y$$
$$x := W(A^T (AWA^T)^{-1} b - c)$$
$$x := \left(A^{-T} B^T B A^{-1} + R^T [\Lambda(Rz)] R\right)^{-1} A^{-T} B^T B A^{-1} y$$

- High-level languages are easy to use, but performance is usually suboptimal.

Introduction

- How to compute the following expressions?

$$b := (X^T X)^{-1} X^T y$$
$$b := (X^T M^{-1} X)^{-1} X^T M^{-1} y$$
$$x := W(A^T (AWA^T)^{-1} b - c)$$
$$x := \left(A^{-T} B^T B A^{-1} + R^T [\Lambda(Rz)] R\right)^{-1} A^{-T} B^T B A^{-1} y$$

- High-level languages are easy to use, but performance is usually suboptimal.
- BLAS and LAPACK can be fast, but require a lot of expertise.

Introduction

- How to compute the following expressions?

$$b := (X^T X)^{-1} X^T y$$
$$b := (X^T M^{-1} X)^{-1} X^T M^{-1} y$$
$$x := W(A^T (AWA^T)^{-1} b - c)$$
$$x := \left(A^{-T} B^T B A^{-1} + R^T [\Lambda(Rz)] R\right)^{-1} A^{-T} B^T B A^{-1} y$$

- High-level languages are easy to use, but performance is usually suboptimal.
- BLAS and LAPACK can be fast, but require a lot of expertise.
- Goal: Simplicity **and** performance.

Introduction

$$b := (X^T X)^{-1} X^T y$$

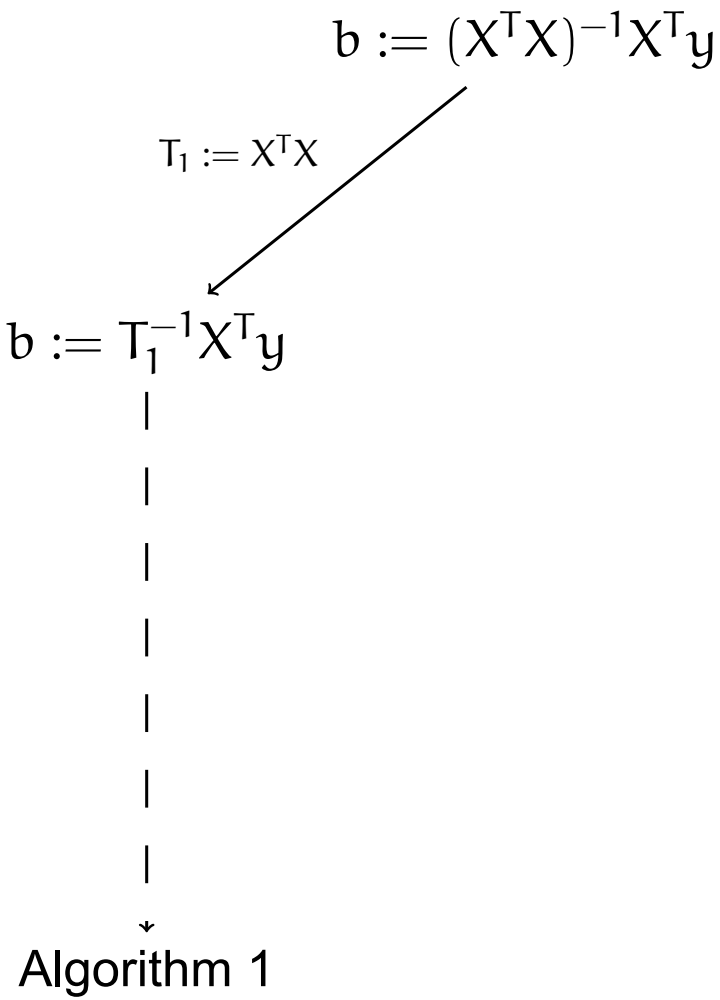
Introduction

$$b := (X^T X)^{-1} X^T y$$

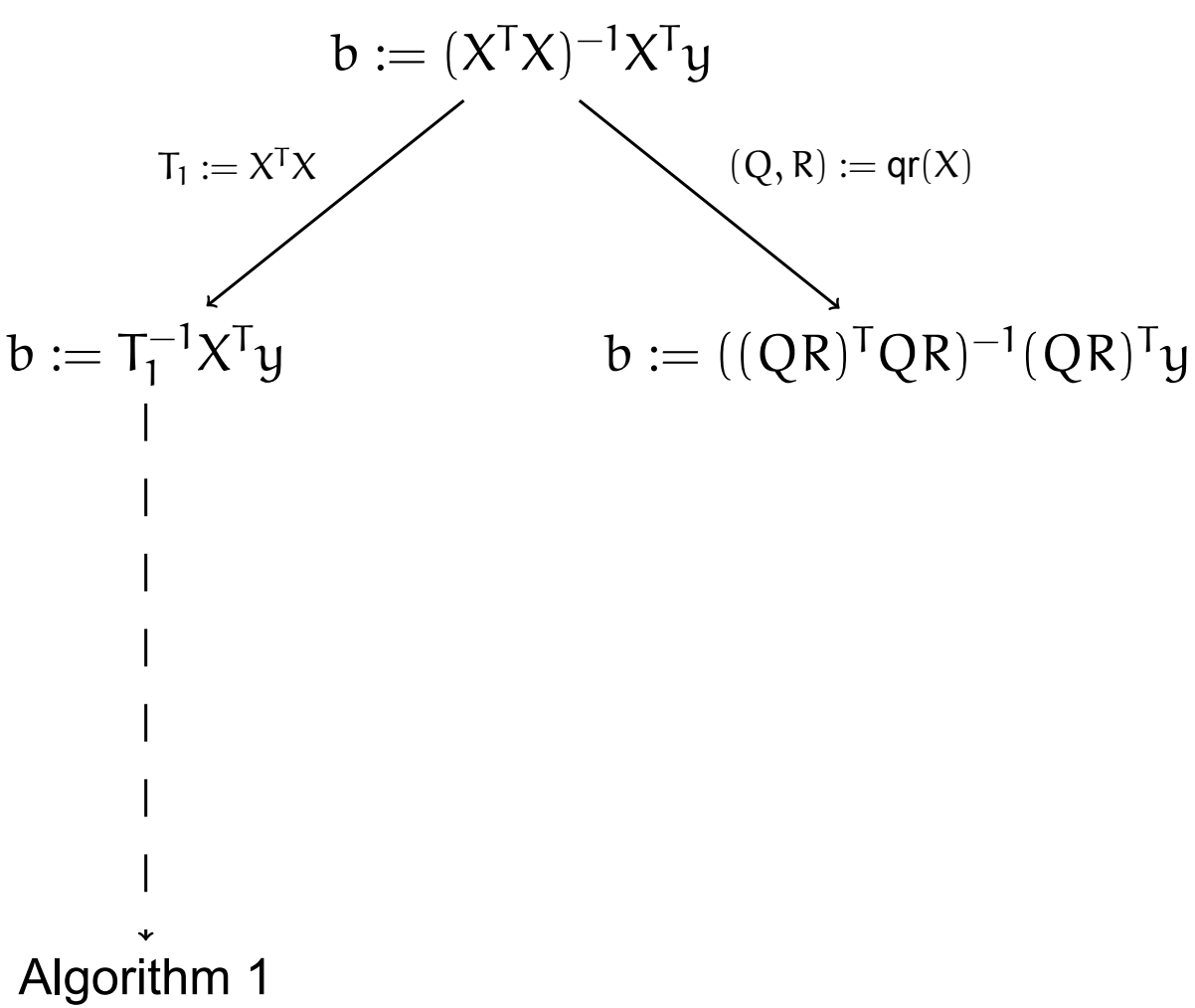
$$T_1 := X^T X$$

$$b := T_1^{-1} X^T y$$

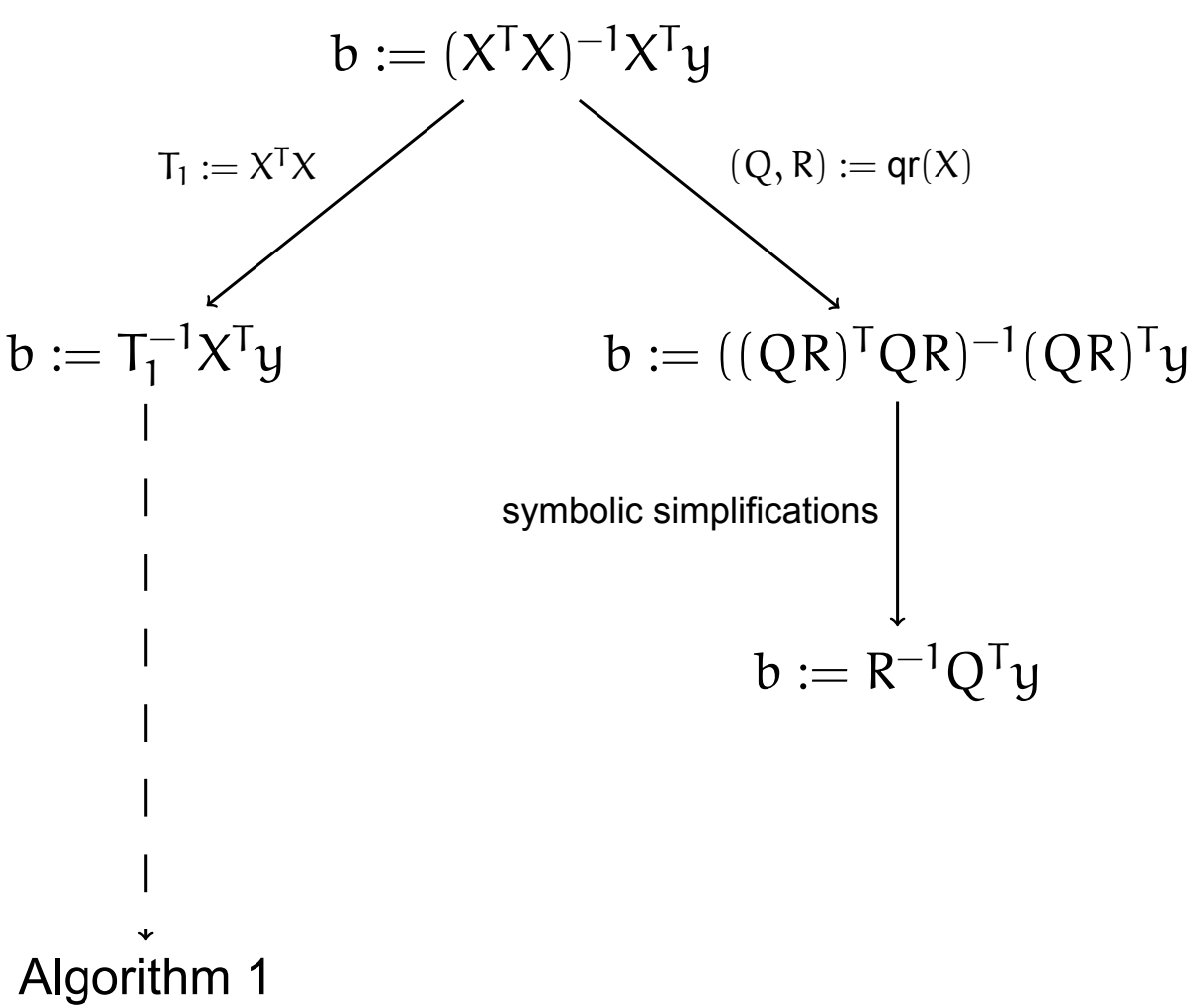
Introduction



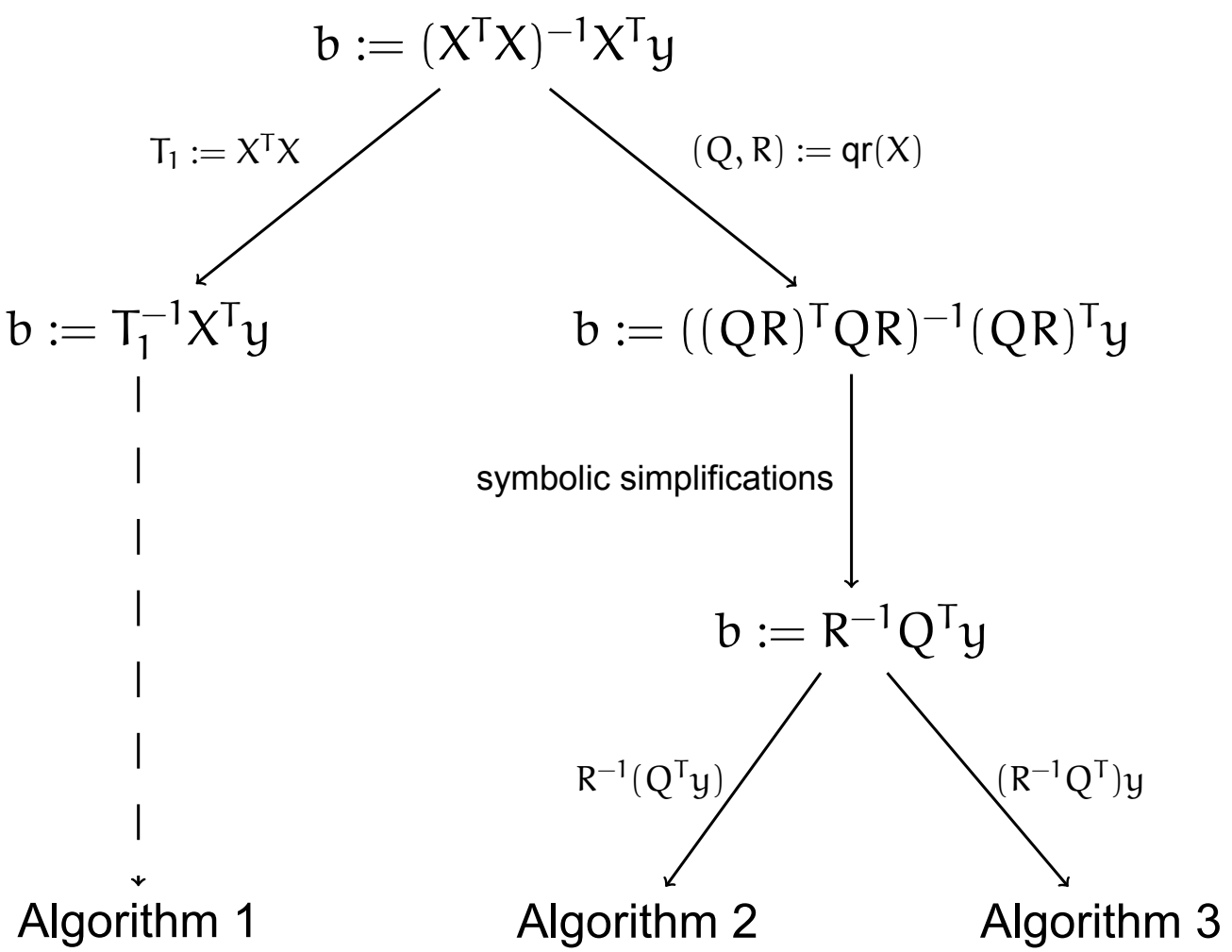
Introduction



Introduction



Introduction



Input

Input

```
n = 1500
m = 1000

Matrix M(n, n) <Input, SPD>
Matrix X(n, m) <Input, FullRank>
ColumnVector y(n) <Input>
ColumnVector b(m) <Output>

b = inv(X'*inv(M)*X)*X'*inv(M)*y
```

Instruction Set

BLAS [DDC+90]

- $y \leftarrow Ax + y$
- $C \leftarrow AB + C$
- $B \leftarrow A^{-1}B$
- ...

LAPACK [AB+99]

- Matrix factorizations.
- Eigensolvers.
- Solvers for linear systems with specific properties.

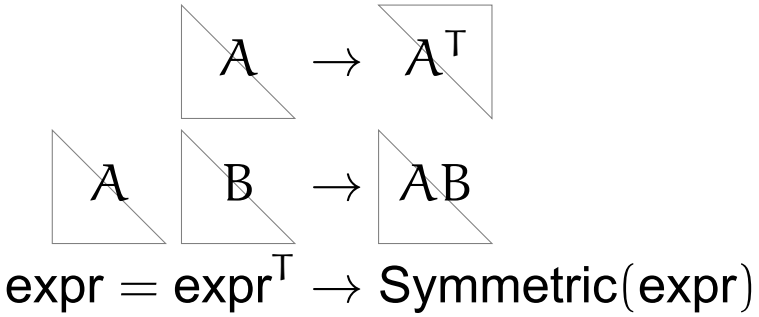
Linear Algebra Knowledge

Linear Algebra Knowledge

Properties

Operation	Cost
A^{-1} B	n^3
A^{-1} B	$2.7n^3$

Inference of Properties



Linear Algebra Knowledge

Simplifications

$$(AB)^T \rightarrow B^T A^T$$

$$A^T \rightarrow A \quad \text{if Symmetric}(A)$$

$$Q^T Q \rightarrow I \quad \text{if Orthogonal}(Q)$$

Rewriting Expressions

$$X := AB + AC \quad \rightarrow \quad X := A(B + C)$$

Rewriting Expressions

$$X := AB + AC \quad \rightarrow \quad X := A(B + C)$$

$$X := A^T A + A^T B + B^T A \quad \rightarrow \quad \begin{aligned} Y &:= B + A/2 \\ X &:= A^T Y + Y^T A \end{aligned}$$

Rewriting Expressions

$$X := AB + AC \quad \rightarrow \quad X := A(B + C)$$

$$X := A^TA + A^TB + B^TA \quad \rightarrow \quad \begin{aligned} Y &:= B + A/2 \\ X &:= A^TY + Y^TA \end{aligned}$$

$$X := (\alpha I + Q^TWQ) \quad \rightarrow \quad \begin{aligned} Y &:= \alpha I + W \\ X &:= Q^TYQ \end{aligned}$$

Linear Algebra Knowledge

Common Subexpression Elimination

$$X := AB^{-T}C$$
$$Y := B^{-1}A^{T}C$$

Common Subexpression Elimination

$$\begin{aligned} X &:= AB^{-T}C \\ Y &:= B^{-1}A^{T}C \end{aligned} \quad \rightarrow \quad \begin{aligned} Z &:= AB^{-T} \\ X &:= ZC \\ Y &:= Z^{T}C \end{aligned}$$

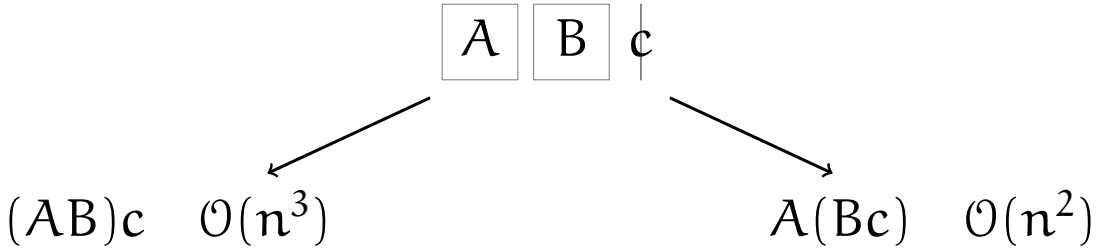
Common Subexpression Elimination

$$\begin{aligned} X &:= AB^{-T}C \\ Y &:= B^{-1}A^{T}C \end{aligned} \quad \rightarrow \quad \begin{aligned} Z &:= AB^{-T} \\ X &:= ZC \\ Y &:= Z^{T}C \end{aligned}$$

$$AB^{-T} = \left(B^{-1}A^{T}\right)^{T} = Z$$

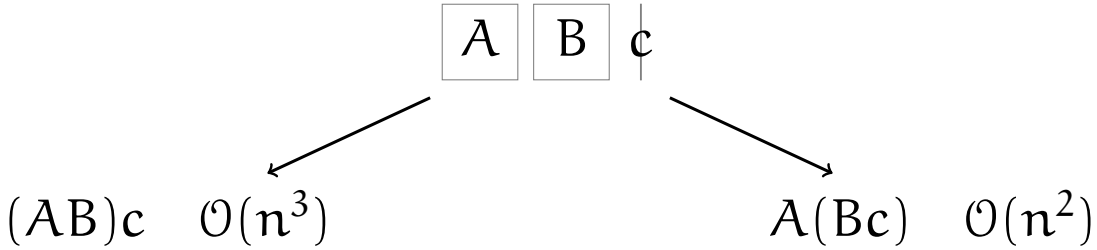
Generalized Matrix Chain Problem

Generalized Matrix Chain Problem



Linear Algebra Knowledge

Generalized Matrix Chain Problem



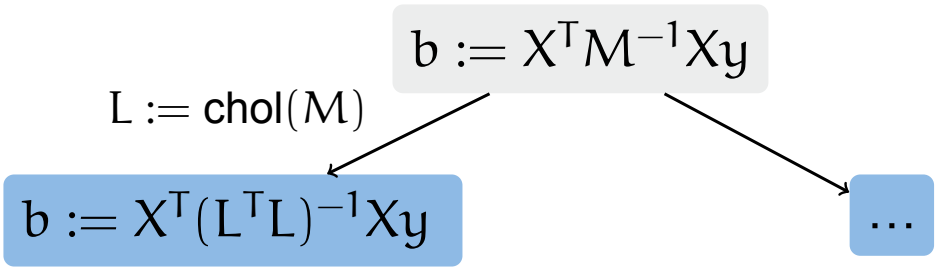
In practice:

$$X := AB^TC^{-T}D$$

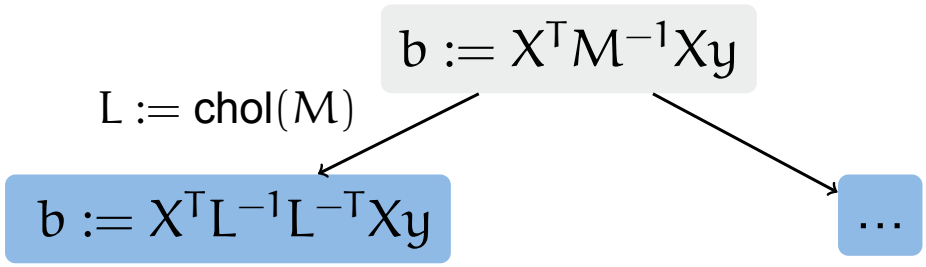
Derivation Graph

$$b := X^T M^{-1} X y$$

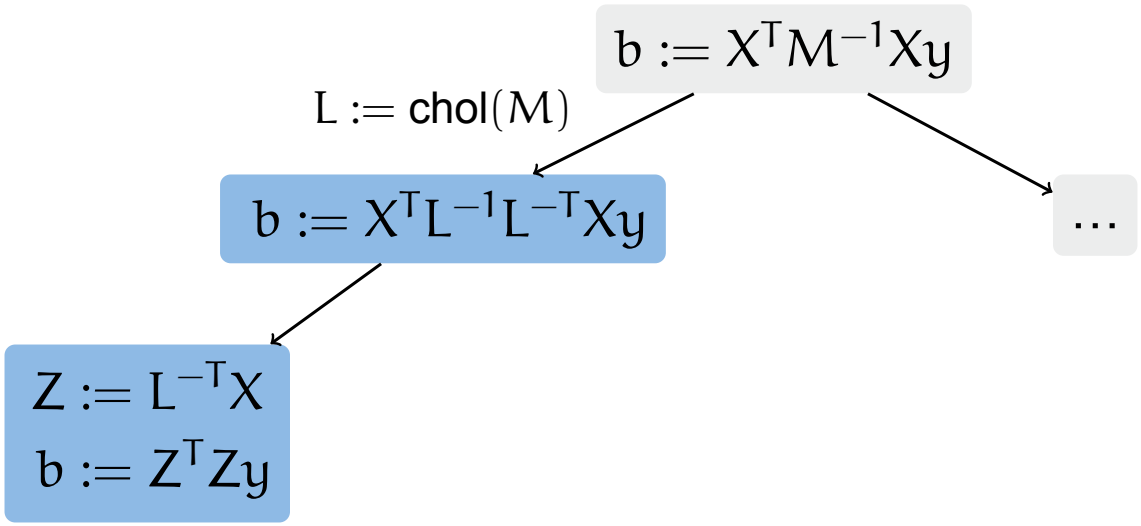
Derivation Graph



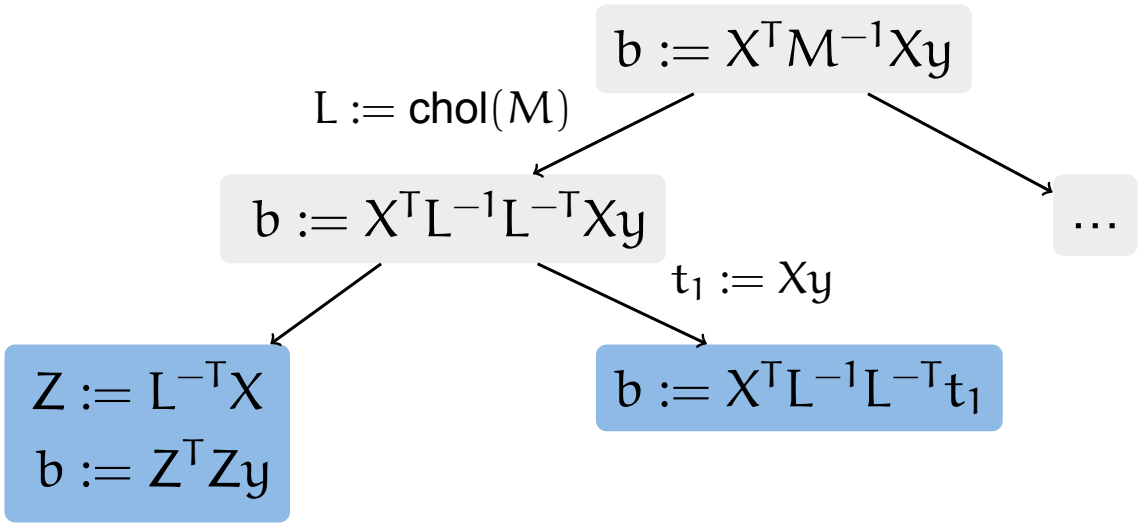
Derivation Graph



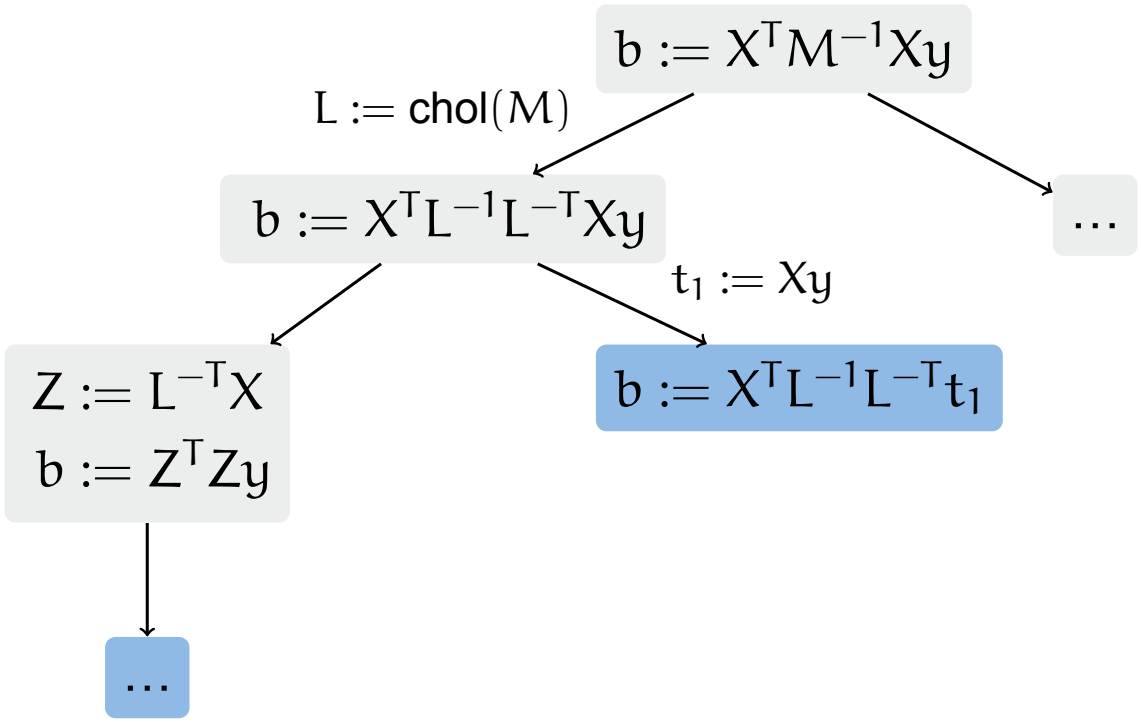
Derivation Graph



Derivation Graph



Derivation Graph



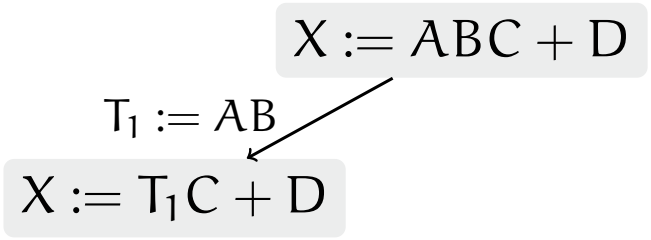
Derivation Graph

Exhaustive

$$X := ABC + D$$

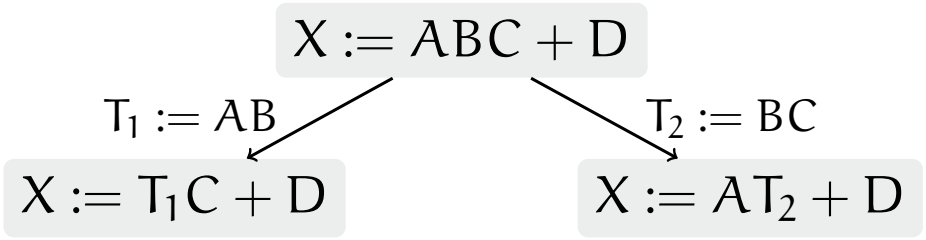
Derivation Graph

Exhaustive



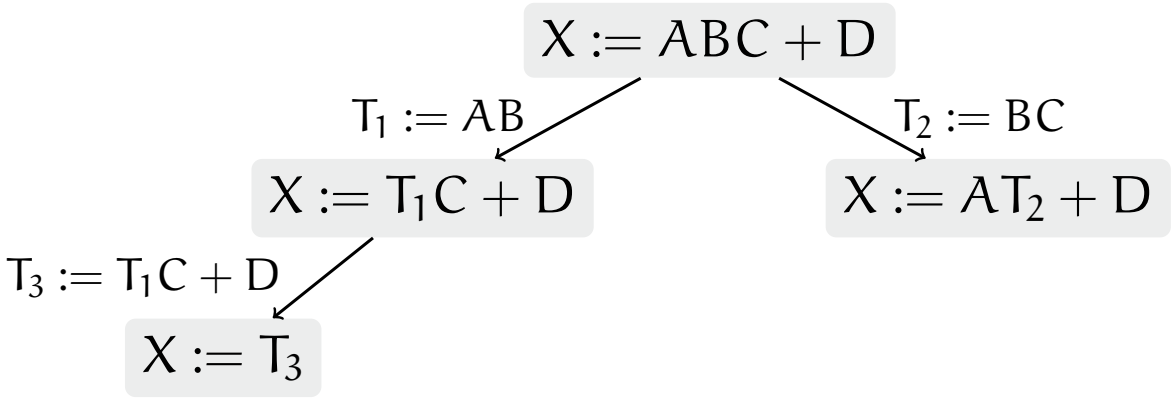
Derivation Graph

Exhaustive



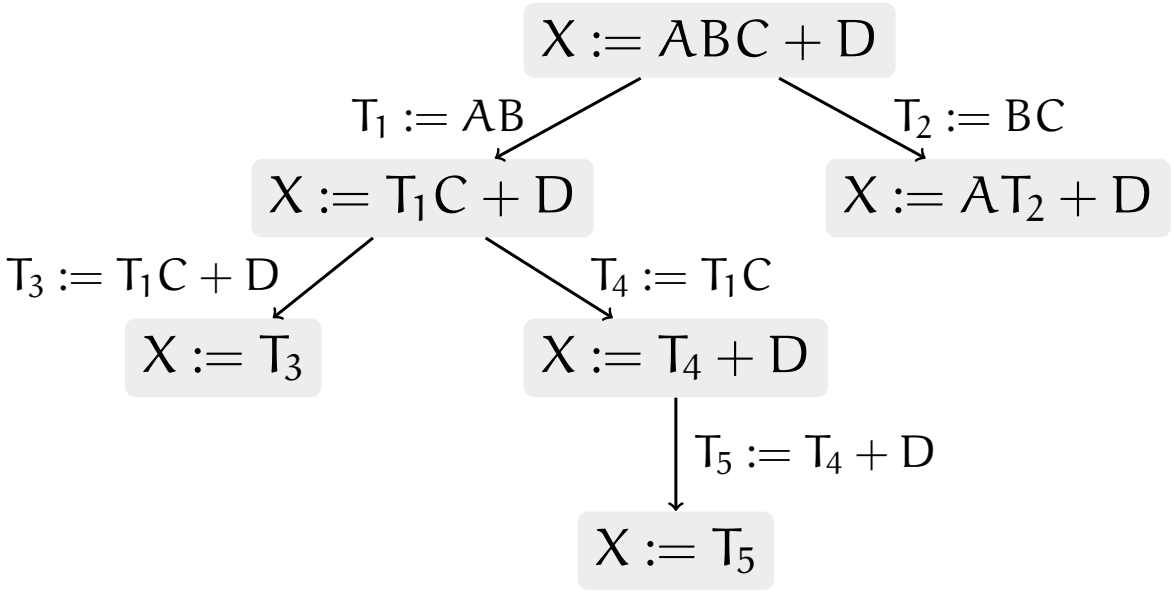
Derivation Graph

Exhaustive



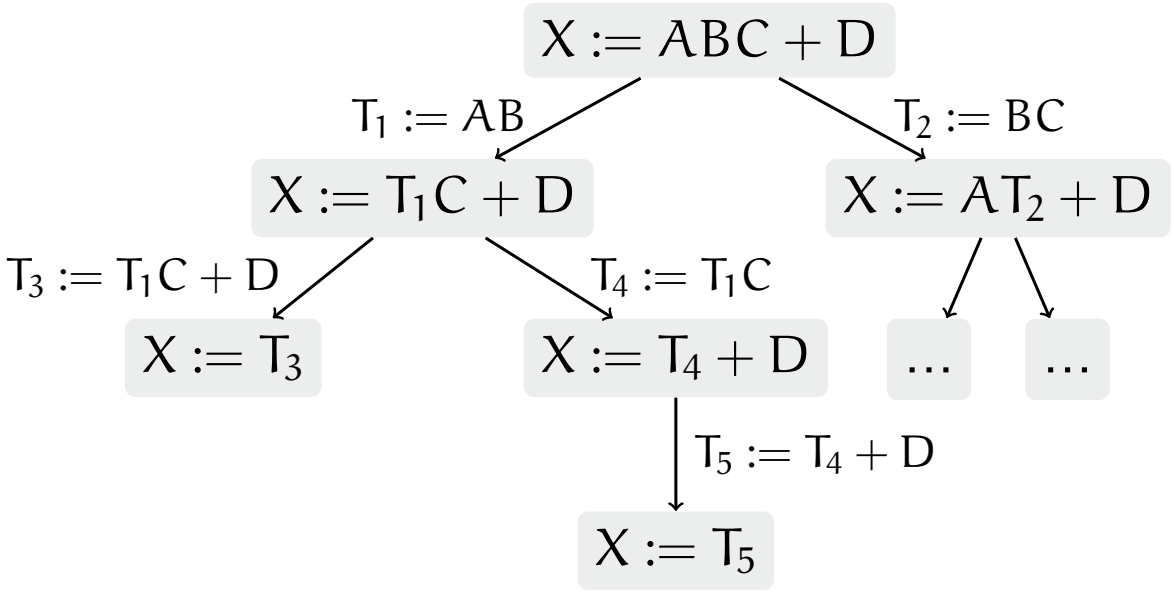
Derivation Graph

Exhaustive



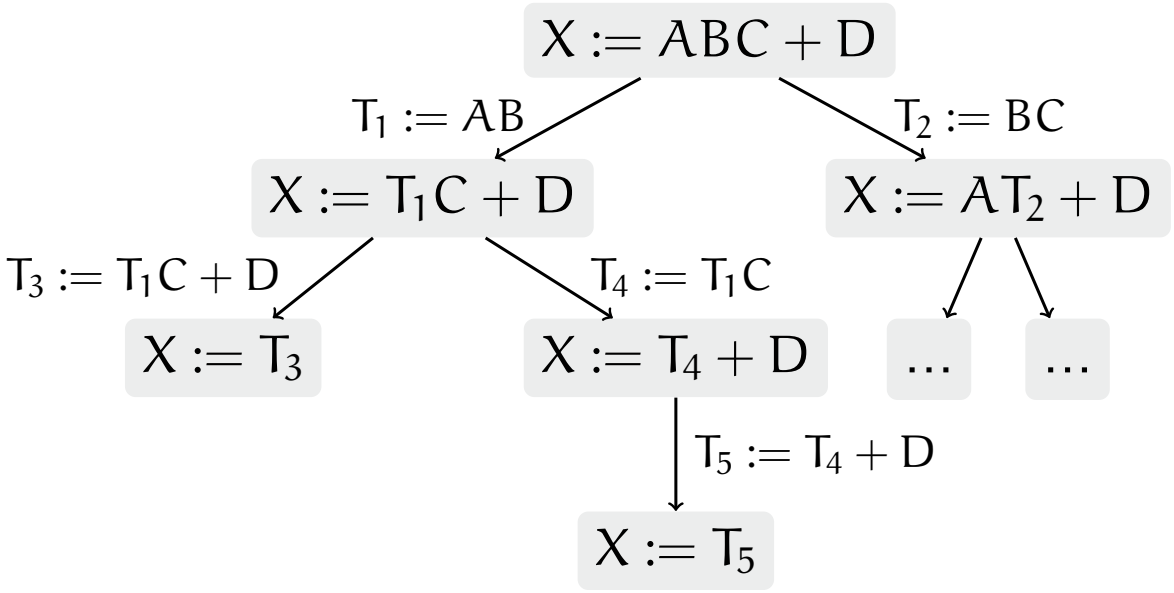
Derivation Graph

Exhaustive



Derivation Graph

Exhaustive

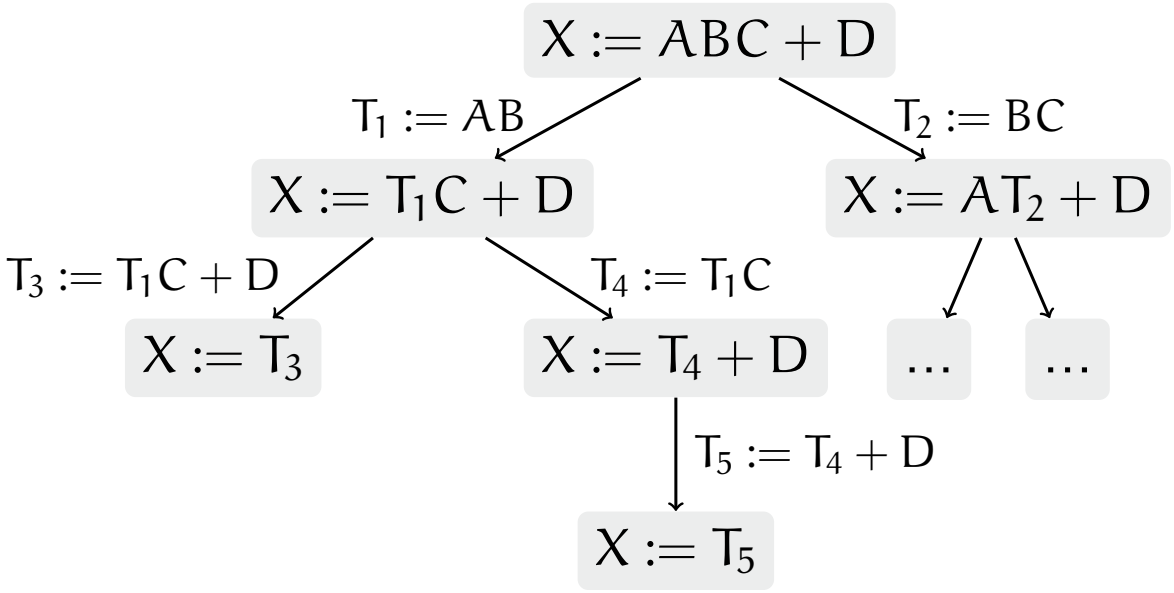


Constructive

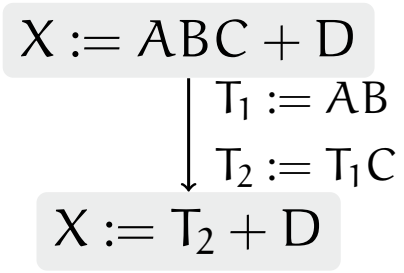
$X := ABC + D$

Derivation Graph

Exhaustive

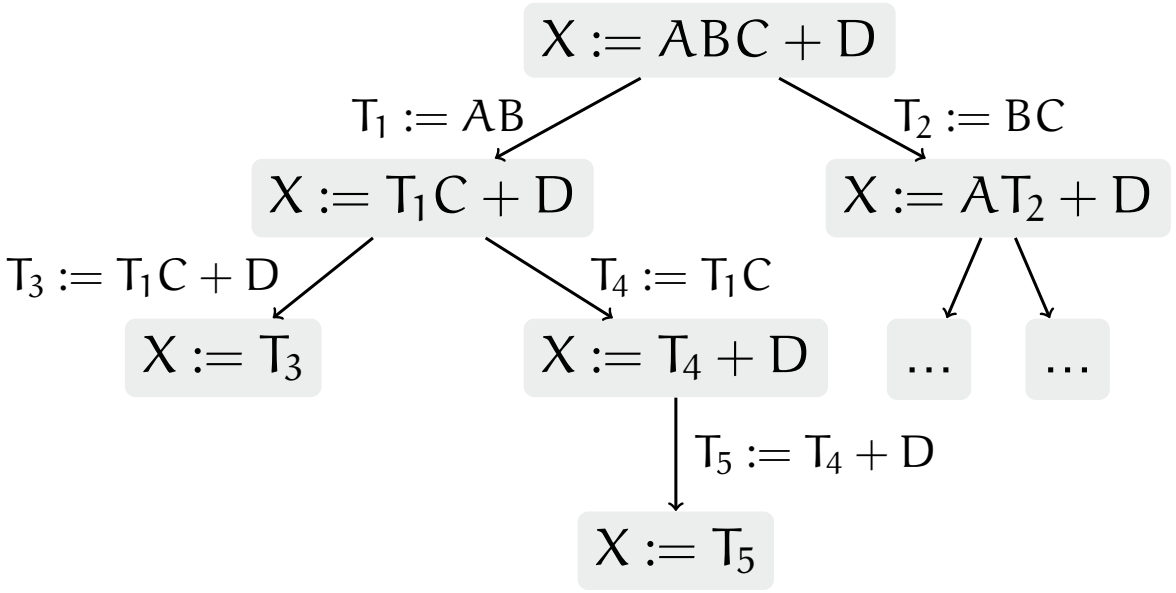


Constructive

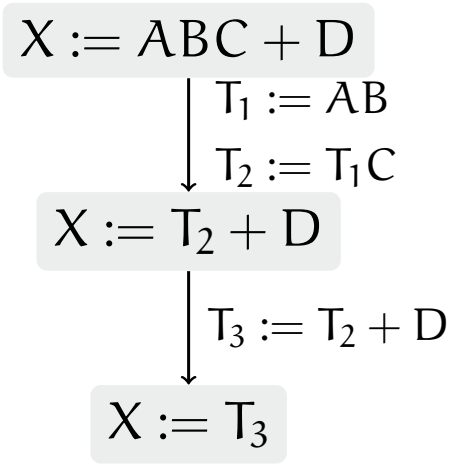


Derivation Graph

Exhaustive



Constructive



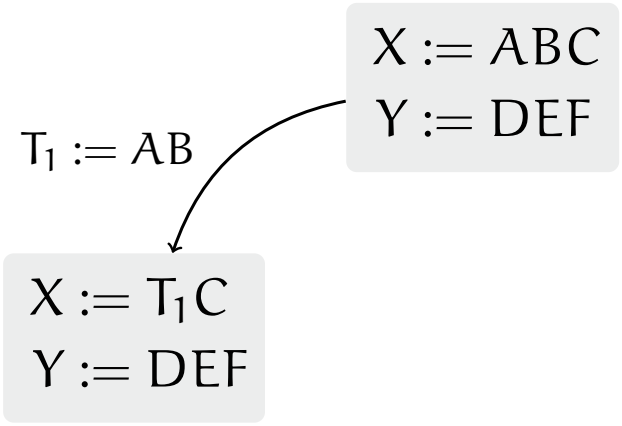
Derivation Graph

Reducing Redundancy

$$X := ABC$$
$$Y := DEF$$

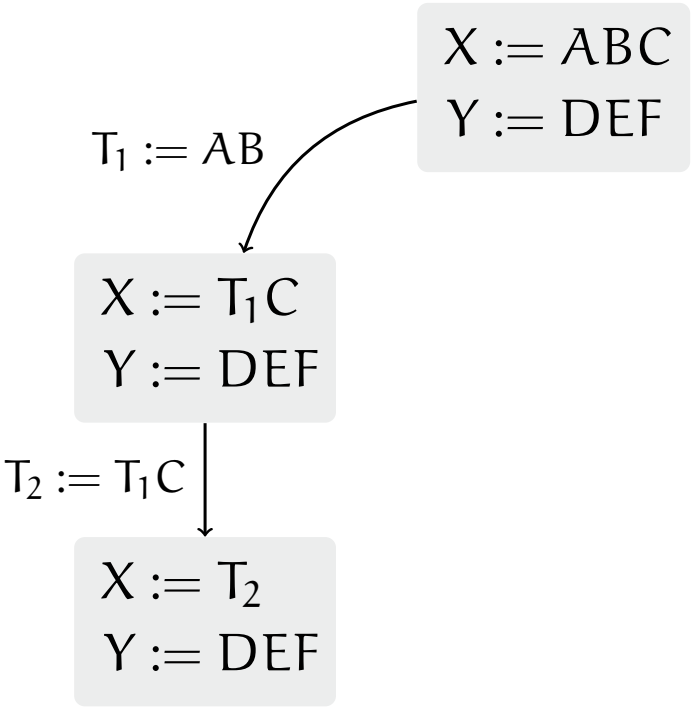
Derivation Graph

Reducing Redundancy



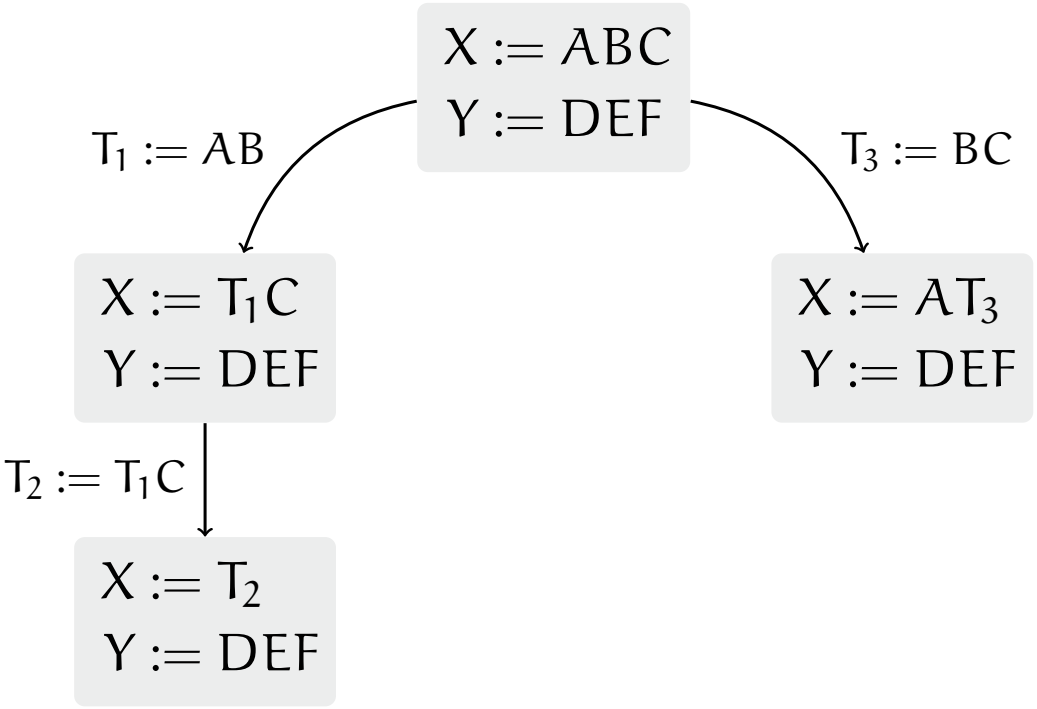
Derivation Graph

Reducing Redundancy



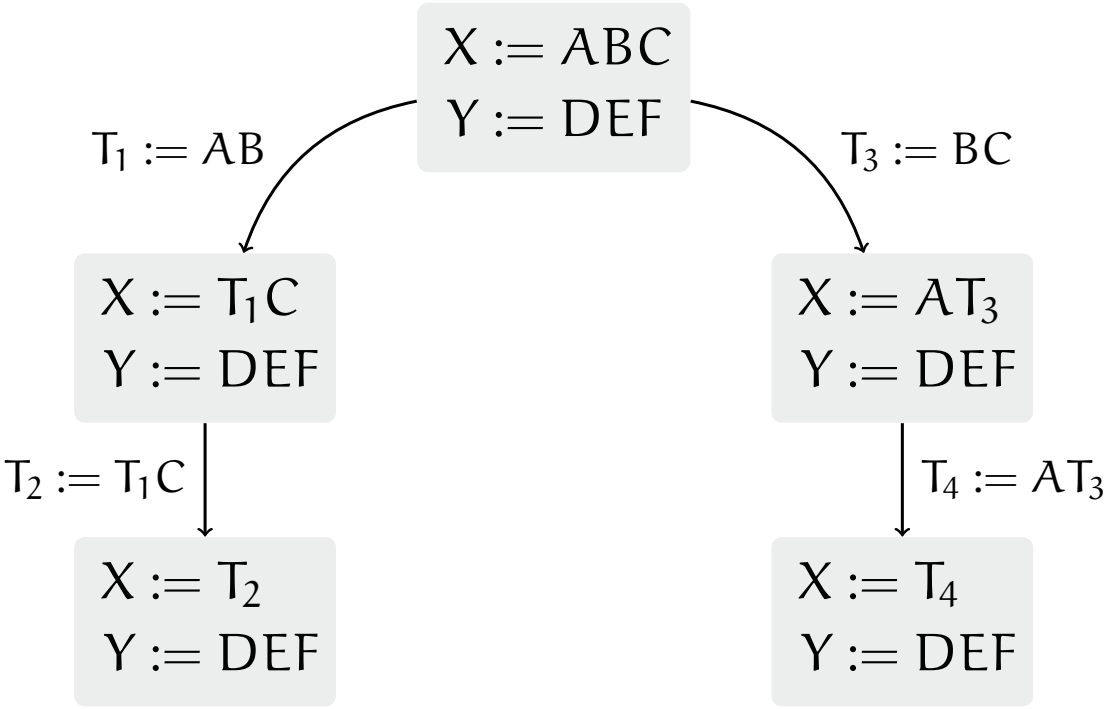
Derivation Graph

Reducing Redundancy

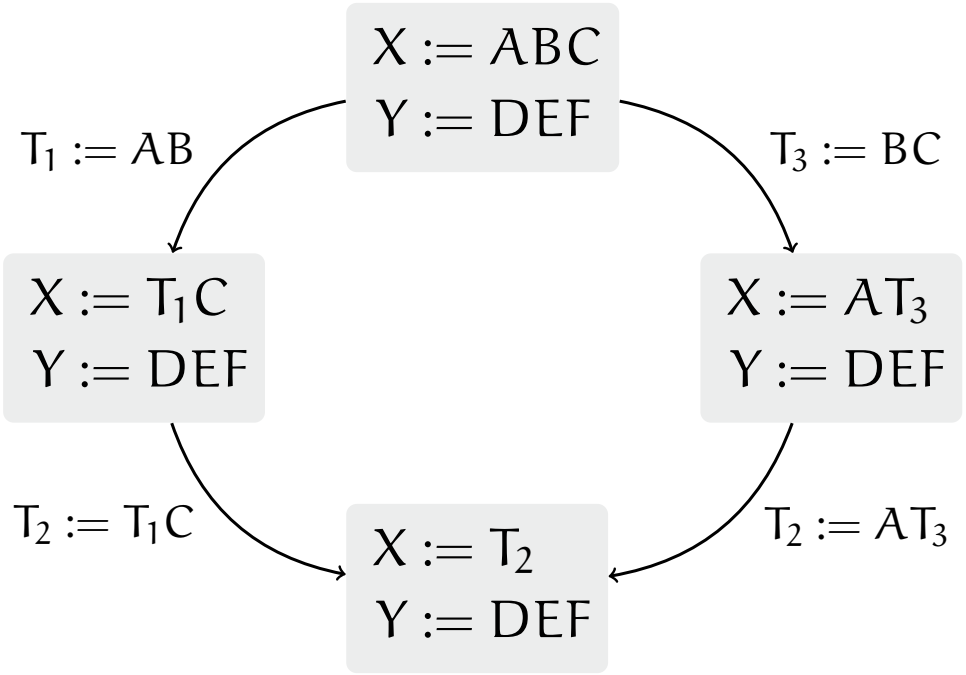


Derivation Graph

Reducing Redundancy



Reducing Redundancy



Results

Example: $w := AB^{-1}c$

Results

Example: $w := AB^{-1}c$

Naive

```
w = A*inv(B)*c
```

Results

Example: $w := AB^{-1}c$

Naive

```
w = A*inv(B)*c
```

Recommended

```
w = A*(B\c)
```

Results

Example: $w := AB^{-1}c$

Naive

```
w = A*inv(B)*c
```

Recommended

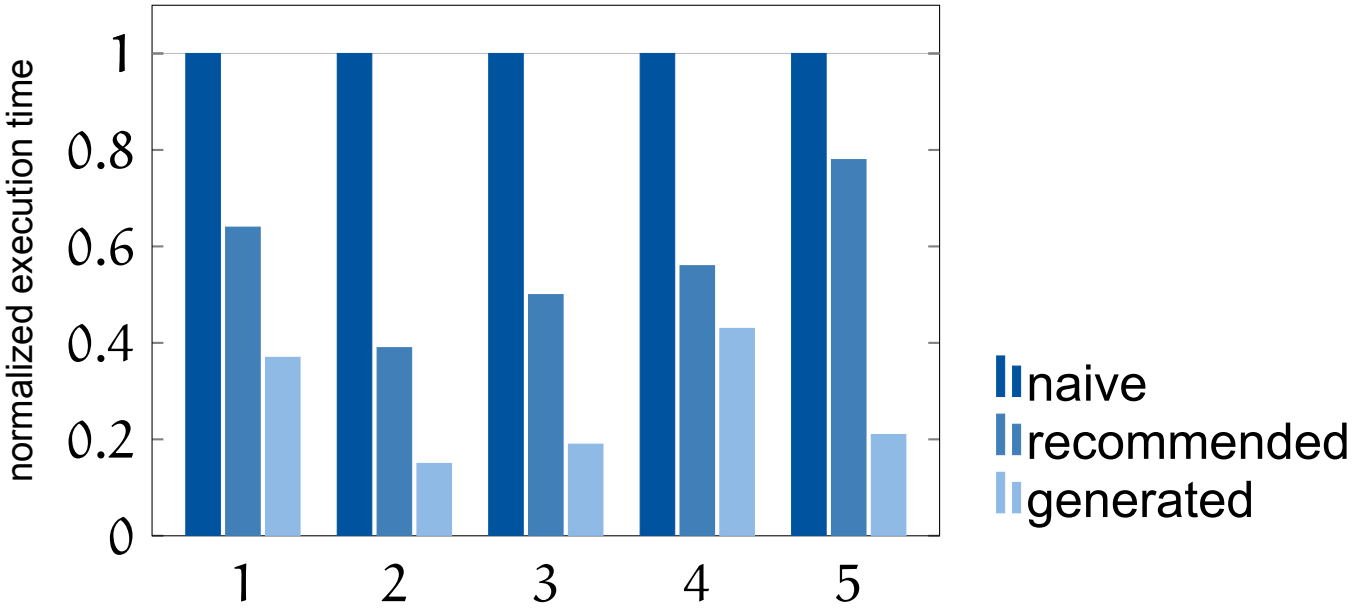
```
w = A*(B\c)
```

Generated

```
ml0 = A; ml1 = B; ml2 = c;
potrf!('L', ml1)
trsv!('L', 'N', 'N', ml1, ml2)
trsv!('L', 'T', 'N', ml1, ml2)
ml3 = Array{Float64}(10)
gemv!('N', 1.0, ml0, ml2, 0.0, ml3)
w = ml3
```

Results

#	Example	code gen time (ms)
1	$b := (X^T X)^{-1} X^T y$	37
2	$b := (X^T M^{-1} X)^{-1} X^T M^{-1} y$	430
3	$W := A^{-1} B C D^{-T} E F$	9
4	$X := AB^{-1}C;\ Y := DB^{-1}A^T$	17
5	$x := W(A^T(AWA^T)^{-1}b - c)$	537



References

[AB+99] Edward Anderson, Zhaojun Bai, et al. *LAPACK Users' guide*, volume 9. SIAM, 1999.

[DDC+90] Jack J. Dongarra, Jeremy Du Croz, et al. A set of Level 3 Basic Linear Algebra Subprograms. *ACM TOMS*, 16(1):1–17, 1990.

Backup

Pattern Matching

Pattern	Subject	Substitution
$f(a, b, x)$	$f(a, b, c)$	$\sigma = \{x \mapsto c\}$
$f_A(a, x)$	$f_A(a, b, c)$	$\sigma = \{x \mapsto f_A(b, c)\}$
$f(g(a, x), y)$	$f(g(a, c), g(a, b))$	$\sigma = \{x \mapsto c, y \mapsto g(a, b)\}$
$f(x^+)$	$f(a, b, c)$	$\sigma = \{x \mapsto [a, b, c]\}$
$f(x, y^*)$	$f(a)$	$\sigma = \{x \mapsto a, y \mapsto \varnothing\}$
$c_1^* X^{-1} Y c_2^*$	$A^{-1}BC$	$\sigma = \{X \mapsto A, Y \mapsto B, c_1 \mapsto \varnothing, c_2 \mapsto [C]\}$

MatchPy: `https://github.com/HPAC/matchpy`

Backup

Inference of Properties

```
def is_lower_triangular(expr):
    if expr is Times:
        return ∀child ∈ expr.children : is_lower_triangular(child)
    if expr is Transpose:
        return is_upper_triangular(expr.child)
    if expr is Matrix:
        return LowerTriangular ∈ expr.properties
    # ...

def is_symmetric(expr):
    if expr is Matrix:
        return Symmetric ∈ expr.properties
    else:
        return expr == transpose(expr)
```