Non-linear Associative-Commutative Many-to-One Pattern Matching with Sequence Variables

Manuel Krebber

June 30, 2017
Table of Contents

1 Introduction

2 Types of Matching

3 Algorithms

4 Experiments

5 Conclusions
Function Symbols:  $f, g$
Term

Function Symbols: $f, g$

Constant Symbols: $a, b, c$
Function Symbols: $f, g$
Constant Symbols: $a, b, c$
Variables: $x, y$
Function Symbols: \( f, g \)
Constant Symbols: \( a, b, c \)
Variables: \( x, y \)

Examples: \( a, f(x, b), f(g(a, b), c) \)
Definition:

Find substitution \( X \) such that \( \hat{\text{pattern}} = \text{subject} \).

Example:

\[
\begin{align*}
\text{Pattern:} & \quad f(x_1, y_2, \ldots, x_n, y_m) \\
\text{Subject:} & \quad f(a, g(b), \ldots, a, y_n, \ldots, g(b))
\end{align*}
\]
Definition: Find substitution $\sigma : \mathcal{X} \to \mathcal{T}$ such that
$\hat{\sigma}(\text{pattern}) = \text{subject}$
**Definition:** Find substitution $\sigma : \mathcal{X} \rightarrow \mathcal{T}$ such that $\hat{\sigma}(\text{pattern}) = \text{subject}$

**Example:** Pattern: $f(x, y)$

\[
\begin{align*}
\text{x} & \mapsto a \\
\text{y} & \mapsto g(b)
\end{align*}
\]

Subject: $f(a, g(b))$
Applications

- Functional programming languages
Applications

- Functional programming languages
- Computer algebra systems (Mathematica)
Applications

- Functional programming languages
- Computer algebra systems (Mathematica)
- Term rewriting systems
Applications

- Functional programming languages
- Computer algebra systems (Mathematica)
- Term rewriting systems
- In this case: Linear Algebra
Table of Contents

1 Introduction

2 Types of Matching

3 Algorithms

4 Experiments

5 Conclusions
Types of Matching

- Syntactic
Types of Matching

- Syntactic
- Linear \((x + y)\) vs. non-linear \((x + x)\)
Types of Matching

- Syntactic
- Linear \((x + y)\) vs. non-linear \((x + x)\)
- Sequence variables
- Associativity
- Commutativity
Types of Matching

- Syntactic
- Linear \((x + y)\) vs. non-linear \((x + x)\)
- Sequence variables
- Associativity
- Commutativity
- Many-to-one vs. one-to-one
$$(A \times B) \times C = A \times (B \times C)$$
\[(A \times B) \times C = A \times (B \times C)\]
Associativity

\[(A \times B) \times C = A \times (B \times C) = A \times B \times C\]

Canonical Variadic Form
Types of Matching

Associativity II

\[ X \times M_3 \]

\[ M_1 \times M_2 \times M_3 \]
Types of Matching

Associativity II

\[ X \times M_3 \]

\[ (M_1 \times M_2) \times M_3 \]

\[ M_1 \times M_2 \times M_3 \]
Types of Matching

Associativity II

\[ X \times M_3 \]

\[ (M_1 \times M_2) \times M_3 \]

\[ M_1 \times M_2 \times M_3 \]
Types of Matching

Associativity II

\[ \sigma = \{ X \mapsto (M_1 \times M_2) \} \]
Sequence Variables

Can match a sequence of terms
Sequence Variables

Can match a sequence of terms

**Notation:** $x^*$ star variable, $x^+$ plus variable

star variables can match empty sequence
Can match a sequence of terms

**Notation:** $x^*$ star variable, $x^+$ plus variable

star variables can match empty sequence

**Example:** $\sigma(f(a, x^+, d)) = f(a, b, c, d)$
Can match a sequence of terms

**Notation:** $x^*$ star variable, $x^+$ plus variable

Star variables can match empty sequence

**Example:**

$\sigma(f(a, x^+, d)) = f(a, b, c, d)$

$\sigma = \{x^+ \mapsto (b, c)\}$
Sequence Variables

Can match a sequence of terms

Notation: $x^*$ star variable, $x^+$ plus variable
star variables can match empty sequence

Example: $\sigma(f(a, x^+, d)) = f(a, b, c, d)$
$\sigma = \{x^+ \mapsto (b, c)\}$

Associativity: $\sigma(f_a(a, x, d)) = f_a(a, b, c, d)$
Can match a sequence of terms

Notation: $x^*$ star variable, $x^+$ plus variable
star variables can match empty sequence

Example: $\sigma(f(a, x^+, d)) = f(a, b, c, d)$
$\sigma = \{x^+ \mapsto (b, c)\}$

Associativity: $\sigma(f_a(a, x, d)) = f_a(a, b, c, d)$
$\sigma = \{x \mapsto f_a(b, c)\}$
Commutativity

$\rightarrow \quad a + b = b + a$
Commutativity

- $a + b = b + a$
- Sort the arguments: $d + a + c + b \rightarrow a + b + c + d$
Commutativity

- \( a + b = b + a \)
- Sort the arguments: \( d + a + c + b \rightarrow a + b + c + d \)
- Every permutation could match: 
  \( n! \) with \( n = |subject| \)
Types of Matching

Complexity

<table>
<thead>
<tr>
<th></th>
<th>Synt</th>
<th>Assoc</th>
<th>SeqVar</th>
<th>Comm</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. # matches</td>
<td>1</td>
<td>( \binom{n-1}{m-1} )</td>
<td>( \binom{n+m-1}{m-1} )</td>
<td>( n! )</td>
<td>( n^m )</td>
</tr>
<tr>
<td>NP complete</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

\[ n = |subject|, \ m = |pattern| \]
Many-to-one Matching

- Many patterns, one subject
Many-to-one Matching

- Many patterns, one subject
- Speedup by simultaneous matching
Many patterns, one subject
Speedup by simultaneous matching
Use similarity between patterns
Table of Contents
Example: Subject $f(a, b)$ and pattern $f(x^*, y^*)$
Example: Subject $f(a, b)$ and pattern $f(x^*, y^*)$

- Try every possible distribution
Example: Subject $f(a, b)$ and pattern $f(x^*, y^*)$

- Try every possible distribution
- Generate (weak) compositions:

\[
\begin{align*}
0 + 2 &= 2 & \{x^* \mapsto (), y^* \mapsto (a, b)\} \\
1 + 1 &= 2 & \{x^* \mapsto (a), y^* \mapsto (b)\} \\
2 + 0 &= 2 & \{x^* \mapsto (a, b), y^* \mapsto ()\}
\end{align*}
\]
Example: Subject $f(a, b)$ and pattern $f(x^*, y^*)$

- Try every possible distribution
- Generate (weak) compositions:

\[
\begin{align*}
0 + 2 &= 2 & \{x^* \mapsto (), y^* \mapsto (a, b)\} \\
1 + 1 &= 2 & \{x^* \mapsto (a), y^* \mapsto (b)\} \\
2 + 0 &= 2 & \{x^* \mapsto (a, b), y^* \mapsto ()\}
\end{align*}
\]

- Backtracking
Steps of Commutative Matching

Pattern: \( f_c(a, g(c, d), g(x), x, y^*, z) \)
Subject: \( f_c(a, b, g(b), c, g(c, d), e) \)
Match: \( \sigma = \{ z \mapsto e \} \)

Permutations: \( 6! = 720 \)
Steps of Commutative Matching

Pattern: $f_c(a, g(c, d), g(x), x, y^*, z)$
Subject: $f_c(a, b, g(b), c, g(c, d), e)$
Match: $\sigma = \{z \mapsto e\}$

1. Constant terms
Steps of Commutative Matching

Pattern: \( f_c(g(x), x, y^*, z) \)

Subject: \( f_c(b, g(b), c, e) \)

Match: \( \sigma = \{ z \mapsto e \} \)

1. Constant terms
2. Matched variables
Steps of Commutative Matching

Pattern: $f_c(g(x), x, y^*)$
Subject: $f_c(b, g(b), c)$
Match: $\sigma = \{z \mapsto e, x \mapsto b\}$

1. Constant terms
2. Matched variables
3. Terms containing variables
Steps of Commutative Matching

Pattern: \( f_c(x, y^*, z) \)
Subject: \( f_c(b, c) \)
Match: \( \sigma = \{ z \mapsto e, x \mapsto b \} \)

1. Constant terms
2. Matched variables
3. Terms containing variables
4. Repeat step 2
Steps of Commutative Matching

Pattern: \( f_c(\quad y^* \quad ) \)
Subject: \( f_c(\quad c \quad ) \)
Match: \( \sigma = \{ z \mapsto e, x \mapsto b \} \)

1. Constant terms
2. Matched variables
3. Terms containing variables
4. Repeat step 2
5. Regular Variables

Permutations: \( 1! = 1 \)
Pattern: $f_c(c, y^*)$

Subject: $f_c(c, y^*)$

Match: $\sigma = \{ z \mapsto e, x \mapsto b, y^* \mapsto \{ c \} \}$

1. Constant terms
2. Matched variables
3. Terms containing variables
4. Repeat step 2
5. Regular Variables
6. Sequence Variables

Permutations: $1! = 1$
Match $f_c(x^+, x^+, y^*)$ and $f_c(a, a, a, b, b, c)$
Match $f_c(x^+, x^+, y^*)$ and $f_c(a, a, a, b, b, c)$

Solve linear diophantine equations (using Extended Euclidean Alg.):

\[
\begin{align*}
3 &= 2x_a + y_a \\
2 &= 2x_b + y_b \\
1 &= 2x_c + y_c \\
1 &\leq x_a + x_b + x_c
\end{align*}
\]
Commutativity + Sequence Variables

Match $f_c(x^+, x^+, y^*)$ and $f_c(a, a, a, b, b, c)$

Solve linear diophantine equations (using Extended Euclidean Alg.):

\[
\begin{align*}
3 &= 2x_a + y_a \\
2 &= 2x_b + y_b \\
1 &= 2x_c + y_c \\
1 &\leq x_a + x_b + x_c
\end{align*}
\]

Solutions:

\[
\begin{align*}
\{x \mapsto \{a, b\}, \quad &y \mapsto \{a, c\}\} \\
\{x \mapsto \{a\}, \quad &y \mapsto \{a, b, b, c\}\} \\
\{x \mapsto \{b\}, \quad &y \mapsto \{a, a, a, c\}\}
\end{align*}
\]
Table of Contents
Patterns: \( f(1, x^*) \), \( f(1) \), \( f(y, 0) \)
Patterns: \( f(1, x^*) \), \( f(1) \), \( f(y, 0) \)

Subject: \( f(1, 0) \)
Patterns: \( f(1, x^*) \), \( f(1) \), \( f(y, 0) \)

Subject: \( f(1, 0) \)
Match: \( f(y, 0) \) with \( \{ y \mapsto 1 \} \)
Patterns: $f(1, x^*), f(1), f(y, 0)$

Subject: $f(1, 0)$
Match: $f(1, x^*)$ with $\{x^* \mapsto 0\}$
Many-to-One for Commutative

Multi Layer Discrimination Net (MLDN)

\[ f_c(a, x, x) \]
\[ f_c(g(b), g(x)) \]
\[ f_c(a, g(x), g(a)) \]
Many-to-One for Commutative

Multi Layer Discrimination Net (MLDN)

\[ f_c(a, g(a), g(a)) \Rightarrow \{1, 1, 1, 2, 2, 4, 4, 5\} \]
1. Match subject arguments with nested DN

Many-to-One for Commutative
Many-to-One for Commutative

1. Match subject arguments with nested DN
2. Build bipartite graph from matches
Many-to-One for Commutative

1. Match subject arguments with nested DN
2. Build bipartite graph from matches
3. Enumerate all maximum matchings (Hopcroft-Karp, Uno)
Many-to-One for Commutative

1. Match subject arguments with nested DN
2. Build bipartite graph from matches
3. Enumerate all maximum matchings (Hopcroft-Karp, Uno)
4. Combine substitutions from matching
1. Match subject arguments with nested DN
2. Build bipartite graph from matches
3. Enumerate all maximum matchings (Hopcroft-Karp, Uno)
4. Combine substitutions from matching
5. Match sequence variables
Bipartite Graph

Subject
\[ f_c(f(a, b), f(a, a)) \]

\( f(a, b) \)

\( f(a, a) \)

Pattern
\[ f_c(f(y, b), f(x, y)) \]

\( f(y, b) \)

\( f(x, y) \)

\( \{ y \mapsto a \} \)

\( \{ x \mapsto a, y \mapsto b \} \)

\( \{ x \mapsto a, y \mapsto a \} \)
Subject
\( f_c(f(a, b), f(a, a)) \)

Pattern
\( f_c(f(y, b), f(x, y)) \)

\( f(a, b) \) \( \{ y \mapsto a \} \) \( f(y, b) \)

\( f(a, a) \) \( \{ x \mapsto a, y \mapsto b \} \) \( f(x, y) \)

\( \{ x \mapsto a, y \mapsto a \} \)
Bipartite Graph

Subject
\[ f_c(f(a, b), f(a, a)) \]

Pattern
\[ f_c(f(a, x), f(x, y)) \]

\{ x \mapsto b \}

\{ x \mapsto a, y \mapsto b \}

\{ x \mapsto a, y \mapsto a \}

\{ x \mapsto b \}

\{ x \mapsto a, y \mapsto b \}

\{ x \mapsto a, y \mapsto a \}
Bipartite Graph

Subject
\( f_c(f(a, b), f(a, a)) \)

Pattern
\( f_c(f(a, x), f(x, y)) \)

\{ \{x \mapsto b\} \}

\{ \{x \mapsto a, y \mapsto b\} \}

\{ \{x \mapsto a, y \mapsto a\} \}
Table of Contents
MatchPy

- Python implementation

Open source on Github: https://github.com/hpac/matchpy
MatchPy

- Python implementation
- Open source on Github:
  https://github.com/hpac/matchpy
~ 200 patterns for BLAS/LAPACK kernels, e.g. $\alpha \times A^T \times B$
~ 200 patterns for BLAS/LAPACK kernels, e.g. $\alpha \times A^T \times B$

- Supported operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol</th>
<th>Arity</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication</td>
<td>$\times$</td>
<td>variadic</td>
<td>associative</td>
</tr>
<tr>
<td>Addition</td>
<td>$+$</td>
<td>variadic</td>
<td>associative, commutative</td>
</tr>
<tr>
<td>Transposition</td>
<td>$T$</td>
<td>unary</td>
<td></td>
</tr>
<tr>
<td>Inversion</td>
<td>$-1$</td>
<td>unary</td>
<td></td>
</tr>
<tr>
<td>Inverse Transposition</td>
<td>$-T$</td>
<td>unary</td>
<td></td>
</tr>
</tbody>
</table>
Match Times

Experiments

Setup Time

Match Time

Match Times

Setup Time [ms]

Match Time [ms]

Number of Patterns

Number of Patterns

match

MTOM
Experiments

Speedup

![Graph showing speedup vs. number of patterns](image1.png)

![Graph showing break even vs. number of patterns](image2.png)

Manuel Krebber | June 30, 2017

28/31
Table of Contents
Contributions

- Generalized discrimination nets
Contributions

- Generalized discrimination nets
- Many-to-one matching
Conclusions

Contributions

- Generalized discrimination nets
- Many-to-one matching
  - sequence variables
Conclusions

Contributions

- Generalized discrimination nets
- Many-to-one matching
  - sequence variables
  - separate associativity/commutativity
Contributions

- Generalized discrimination nets
- Many-to-one matching
  - sequence variables
  - separate associativity/commutativity
- Open source implementation
Project is on GitHub:
https://github.com/hpac/matchpy
Thank you for your attention!