Properties of the Generalized Matching Algorithm

Maximilian Donnermair

2024-04-23

Problem statement

(Proximity-based) Matching

Given:

- a proximity relation $\mathcal{R}$
- a cut value λ
- two terms t and s

Find: all $(\mathcal{R}, \lambda)$-matchers of t to s, i.e. substitutions σ such that $\mathcal{R}(t\sigma, s) \geq \lambda$.

Problem statement

(Proximity-based) Matching

Given:

- a proximity relation $\mathcal{R}$
- a cut value λ
- two terms t and s

Find: all $(\mathcal{R}, \lambda)$-matchers of t to s, i.e. substitutions σ such that $\mathcal{R}(t\sigma, s) \geq \lambda$.

For computing proximity degrees of terms, **T-Norms** are used.

$$\mathcal{R}(f(t_1, \ldots, t_n), g(s_1, \ldots, s_n)) =$$
$$\mathcal{R}(f, g) \otimes \mathcal{R}(t_1, s_1) \otimes \ldots \otimes \mathcal{R}(t_n, s_n)$$

Problem statement

(Proximity-based) Matching

Given:

- a proximity relation $\mathcal{R}$
- a cut value λ
- two terms t and s

Find: all $(\mathcal{R}, \lambda)$-matchers of t to s, i.e. substitutions σ such that $\mathcal{R}(t\sigma, s) \geq \lambda$.

For computing proximity degrees of terms, **T-Norms** are used.
$\mathcal{R}(f(t_1, \ldots, t_n), g(s_1, \ldots, s_n)) =$
$\mathcal{R}(f, g) \otimes \mathcal{R}(t_1, s_1) \otimes \ldots \otimes \mathcal{R}(t_n, s_n)$
State of the art: Proximity-based matching algorithms for $t \otimes s = min(t, s)$ (Gödel- or Minimum-T-Norm)

Problem statement

Why are general T-Norms so different to the Minimum-Norm?

Example

L $\mathcal{R} = \{a \approx_{0.8} c \approx_{0.8} b, a \approx_{0.9} d \approx_{0.9} b\}$, we match
$f(x, y) \preceq f(a, b)$ with $0.75 = \lambda$-cut.
We can see that $x \mapsto c$ matches a with proximity degree 0.8 and $y \mapsto d$ matches b with proximity degree 0.9. In the case of the Minimum-T-Norm, these two can be viewed independently from each other. With general T-Norms, both substitutions depend on each other.

Inference system $\mathfrak{M}$

A set of rewrite rules that works on tuples of the form $M; S; D$.

- $M := \{t \preceq_\delta s\}$ (matching problems)
- $S := \{x \approx \mathbf{r}\}$ (variable constraints)
- $D \geq \lambda$ (constraint factor)

Inference system $\mathfrak{M}$

A set of rewrite rules that works on tuples of the form $M; S; D$.

- $M := \{t \preceq_\delta s\}$ (matching problems)
- $S := \{x \approx \mathbf{r}\}$ (variable constraints)
- $D \geq \lambda$ (constraint factor)

$M; \emptyset; 1 \Longrightarrow^+ \emptyset; S; D$, where
S and D form constraints which are met by substitutions iff they are a $(\mathcal{R}, \lambda)$-matcher of the original problem.
Thus, constraint solving is also part of the algorithm.

Inference system $\mathfrak{M}$

A set of rewrite rules that works on tuples of the form $M; S; D$.

- $M := \{t \preceq_\delta s\}$ (matching problems)
- $S := \{x \approx \mathbf{r}\}$ (variable constraints)
- $D \geq \lambda$ (constraint factor)

$M; \emptyset; 1 \Longrightarrow^+ \emptyset; S; D$, where
S and D form constraints which are met by substitutions iff they are a $(\mathcal{R}, \lambda)$-matcher of the original problem.
Thus, constraint solving is also part of the algorithm.

Alternatively: $M; \emptyset; 1 \Longrightarrow^+ \perp$ if unsatisfiability is detected early on.

Inference system $\mathfrak{M}$

A set of rewrite rules that works on tuples of the form $M; S; D$.

- $M := \{t \preceq_\delta s\}$ (matching problems)
- $S := \{x \approx \mathbf{r}\}$ (variable constraints)
- $D \geq \lambda$ (constraint factor)

$\mathbf{r}$ denotes a graded set of terms, i.e. $\{(r, \alpha_r) \mid r \in \mathcal{T} \wedge \alpha_r \in [0, 1]\}$.

Inference system $\mathfrak{M}$

A set of rewrite rules that works on tuples of the form $M; S; D$.

- $M := \{t \preceq_\delta s\}$ (matching problems)
- $S := \{x \approx \mathbf{r}\}$ (variable constraints)
- $D \geq \lambda$ (constraint factor)

$\mathbf{r}$ denotes a graded set of terms, i.e. $\{(r, \alpha_r) \mid r \in \mathcal{T} \land \alpha_r \in [0, 1]\}$.

If $\mathbf{r}$ is a proximity class $\mathbf{pc}_{\mathcal{R},\delta}(r')$ of a term r, then it equals $\{(r, \alpha_r) \mid \alpha_r = \mathcal{R}(r, r') \geq \delta\}$.

Inference system $\mathfrak{M}$

A set of rewrite rules that works on tuples of the form $M; S; D$.

- $M := \{t \preceq_\delta s\}$ (matching problems)
- $S := \{x \approx \mathbf{r}\}$ (variable constraints)
- $D \geq \lambda$ (constraint factor)

$\mathbf{r}$ denotes a graded set of terms, i.e. $\{(r, \alpha_r) \mid r \in \mathcal{T} \wedge \alpha_r \in [0,1]\}$.

If $\mathbf{r}$ is a proximity class $\mathbf{pc}_{\mathcal{R},\delta}(r')$ of a term r, then it equals $\{(r, \alpha_r) \mid \alpha_r = \mathcal{R}(r, r') \geq \delta\}$.

The intersection of such sets is defined as $\mathbf{pc}_{\mathcal{R},\delta_1}(t) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(s) := \{(r, \alpha_r) \mid \exists_{\substack{(p,\alpha_p)\in\mathbf{pc}_{\mathcal{R},\delta_1}(t)\\(q,\alpha_q)\in\mathbf{pc}_{\mathcal{R},\delta_2}(s)}} : p = q = r \wedge \alpha_r = \alpha_p \otimes \alpha_q\}$

Rules

Decomposition

$$\{f(t_1,\dots,t_n) \preceq_\delta g(s_1,\dots,s_n)\} \uplus M; S; D \Longrightarrow$$
$$M \cup \{t_i \preceq_{\delta_i} s_i \mid 1 \le i \le n\}; S; D \otimes \mathcal{R}(f,g),$$

if $\mathcal{R}(f,g) \ge \lambda$.

Rules

Decomposition

$$\{f(t_1,\dots,t_n) \preceq_\delta g(s_1,\dots,s_n)\} \uplus M; S; D \Longrightarrow$$
$$M \cup \{t_i \preceq_{\delta_i} s_i \mid 1 \leq i \leq n\}; S; D \otimes \mathcal{R}(f,g),$$

if $\mathcal{R}(f,g) \geq \lambda$.

Dec-Clash

$$\{f(t_1,\dots,t_n) \preceq_\delta g(s_1,\dots,s_m)\} \uplus M; S; D \Longrightarrow \bot$$

if $n \neq m$ or $\mathcal{R}(f,g) < \lambda$

Rules

Decomposition

$$\{f(t_1,\dots,t_n) \preceq_\delta g(s_1,\dots,s_n)\} \uplus M; S; D \Longrightarrow$$
$$M \cup \{t_i \preceq_{\delta_i} s_i \mid 1 \le i \le n\}; S; D \otimes \mathcal{R}(f,g),$$

if $\mathcal{R}(f,g) \ge \lambda$.

Dec-Clash

$$\{f(t_1,\dots,t_n) \preceq_\delta g(s_1,\dots,s_m)\} \uplus M; S; D \Longrightarrow \bot$$

if $n \neq m$ or $\mathcal{R}(f,g) < \lambda$

Solve

$$\{x \preceq_\delta t\} \uplus M; S; D \Longrightarrow M; S \cup \{x \approx \mathbf{pc}_{\mathcal{R},\delta}(t)\}; D$$

Rules

Decomposition

$$\{f(t_1,\ldots,t_n) \preceq_\delta g(s_1,\ldots,s_n)\} \uplus M; S; D \Longrightarrow$$
$$M \cup \{t_i \preceq_{\delta_i} s_i \mid 1 \leq i \leq n\}; S; D \otimes \mathcal{R}(f,g),$$

if $\mathcal{R}(f,g) \geq \lambda$.

Dec-Clash

$$\{f(t_1,\ldots,t_n) \preceq_\delta g(s_1,\ldots,s_m)\} \uplus M; S; D \Longrightarrow \bot$$

if $n \neq m$ or $\mathcal{R}(f,g) < \lambda$

Solve

$$\{x \preceq_\delta t\} \uplus M; S; D \Longrightarrow M; S \cup \{x \approx \mathbf{pc}_{\mathcal{R},\delta}(t)\}; D$$

Merge

$$M; S \uplus \{x \approx \mathbf{t}, x \approx \mathbf{s}\}; D \Longrightarrow M; S \cup \{x \approx \mathbf{t} \sqcap \mathbf{s}\}; D$$

Rules: Remarks

Early failure detection

The Clash rule for the case $R(f, g) < \lambda$ during Decomposition, which allows us to stop the algorithm prematurely, is not needed for proving correctness, because the failure would be detected during constraint solving anyway.
However, it is important for efficiency.

Rules: Remarks

Early failure detection
The Clash rule for the case $R(f, g) < \lambda$ during Decomposition, which allows us to stop the algorithm prematurely, is not needed for proving correctness, because the failure would be detected during constraint solving anyway.
However, it is important for efficiency.

There are other cases where the inference rules can be refined.

Rules: Remarks

Early failure detection
The Clash rule for the case $R(f, g) < \lambda$ during Decomposition, which allows us to stop the algorithm prematurely, is not needed for proving correctness, because the failure would be detected during constraint solving anyway.
However, it is important for efficiency.

There are other cases where the inference rules can be refined.

Merging: Also the Merge rule is technically not necessary, since it is only a different way of stating how close x has to be to which terms.
It helps however in avoiding blowups in term and set representation.

Termination

Termination of a rule-based system can be shown by

- defining a well-founded ordering on the expressions the system operates on, and
- proving that each rule strictly decreases the ordering.

Termination

Termination of a rule-based system can be shown by

- defining a well-founded ordering on the expressions the system operates on, and
- proving that each rule strictly decreases the ordering.

Definition

With

- $\mathit{size}(t)$: the number of symbols in a term t,
- $\mathit{size}(M) := \sum_{t \preceq s \in M} (\mathit{size}(t) + \mathit{size}(s))$,
- $|S|$ is the cardinality of S, i.e. the number of equations of the form $x \approx \mathbf{r}$,

Termination

Termination of a rule-based system can be shown by

- defining a well-founded ordering on the expressions the system operates on, and
- proving that each rule strictly decreases the ordering.

Definition

With

- $size(t)$: the number of symbols in a term t,
- $size(M) := \sum_{t \preceq s \in M} (size(t) + size(s))$,
- $|S|$ is the cardinality of S, i.e. the number of equations of the form $x \approx \mathbf{r}$,

the ordering $\triangleright$ for our systems is defined as:

$$M; S; D \triangleright M'; S'; D' \text{ iff } size(M) + |S| > size(M') + |S'|$$

Termination

The ordering $\rhd$ is obviously well-founded.

For each rule performing $M; S; D \Longrightarrow M'; S'; D'$, we have $M; S; D \rhd M'; S'; D'$ because

- Decomposition reduces the size of M without affecting S,
- Solve increases $|S|$ by one, but decreases the size of M by at least two,
- Merge decreases $|S|$ without affecting M.

Example

We solve $\{f(x,x) \preceq g(f(a,b),h(c,d))\}$ with $\lambda = 0.3$ and $\mathcal{R} :=$

- $\{f \approx_{0.9} g, g \approx_{0.8} h, h \approx_{0.25} f\} \cup$
- $\{a \approx_{0.95} b, b \approx_{0.75} c, c \approx_{0.85} d, d \approx_{0.82} a\}$

Steps:

$$\{f(x,x) \preceq_\delta g(f(a,b),h(c,d))\}; \emptyset; 1 \Longrightarrow_{DEC}$$
$$\{x \preceq_{\delta_1} f(a,b), x \preceq_{\delta_2} h(c,d)\}; \emptyset; 1 \otimes \mathcal{R}(f,g) \Longrightarrow_{SOL}$$
$$\{x \preceq_{\delta_2} h(c,d)\}; \{x \approx \mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b))\}; 1 \otimes \mathcal{R}(f,g) \Longrightarrow_{SOL}$$
$$\emptyset; \{x \approx \mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)), x \approx \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))\};$$
$$1 \otimes \mathcal{R}(f,g) \Longrightarrow_{MER}$$
$$\emptyset; \{x \approx \mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))\};$$
$$1 \otimes \mathcal{R}(f,g)$$

Defining solutions

Definition

A substitution σ is an $(\mathcal{R}, \lambda)$-matcher of $M; S; D$ iff the following conditions hold:

1. σ is an $(\mathcal{R}, \lambda)$-matcher of M under D and S, i.e.
   $\bigotimes_{t \preceq s \in M} \mathcal{R}(t\sigma, s) \bigotimes_{x \approx \mathbf{r}_\delta \in S} \delta \otimes D \geq \lambda$
2. for all $(x \approx \mathbf{r}_\delta) \in S$, we have $\bigvee_{(r, \alpha_r) \in \mathbf{r}_\delta} (x\sigma = r) \wedge (\alpha_r = \delta)$

Defining solutions

Definition

A substitution σ is an $(\mathcal{R}, \lambda)$-matcher of $M; S; D$ iff the following conditions hold:

1. σ is an $(\mathcal{R}, \lambda)$-matcher of M under D and S, i.e.
   $$\bigotimes_{t \preceq s \in M} \mathcal{R}(t\sigma, s) \bigotimes_{x \approx \mathbf{r}_\delta \in S} \delta \otimes D \geq \lambda$$
2. for all $(x \approx \mathbf{r}_\delta) \in S$, we have $\bigvee_{(r, \alpha_r) \in \mathbf{r}_\delta} (x\sigma = r) \wedge (\alpha_r = \delta)$

This definition coincides for the initial step $M; \emptyset; 1$ with the definition of a matcher of the original problem.

Defining solutions

Lemma

If $M_1; S_1; D_1 \implies M_2; S_2; D_2$ is a step of the generalized matching algorithm, then σ is a matcher of $M_1; S_1; D_1$ iff it is a matcher of $M_2; S_2; D_2$.

Proof. A step of the algorithm is an application of one of the rules, thus it has to hold for each individually.

- Dec: $S_1 = S_2$. Let w.l.o.g.
  $M_1 = t := f(t_1, \ldots, t_n) \preceq s := g(s_1, \ldots, s_n)$, thus
  $M_2 = \{t_1 \preceq s_1, \ldots, t_n \preceq s_n\}$.

  Since $D_2 = D_1 \otimes \mathcal{R}(f, g)$, we get
  $D_1 \otimes \mathcal{R}(t\sigma, s) \geq \lambda \iff D_2 \otimes \bigotimes_{1 \leq i \leq n} \mathcal{R}(t_i\sigma, s_i) \geq \lambda.$

Defining solutions

Lemma

If $M_1; S_1; D_1 \implies M_2; S_2; D_2$ is a step of the generalized matching algorithm, then σ is a matcher of $M_1; S_1; D_1$ iff it is a matcher of $M_2; S_2; D_2$.

Proof. A step of the algorithm is an application of one of the rules, thus it has to hold for each individually.

- Sol: $D_2 = D_1$ and let $M_1 := \{x \preceq_\delta t\}$ and $S_1 := \emptyset$. Now $S_2 = \{x \approx \mathbf{pc}_{\mathcal{R},\delta}(t)\}$ implies for a matcher σ that $\exists_{(r,\alpha)}$ with $\alpha = \mathcal{R}(x\sigma = r, t) = \delta$ and thus

$$D_1 \otimes \mathcal{R}(x\sigma, t) \geq \lambda \iff D_2 \otimes \delta \geq \lambda.$$

Defining solutions

Lemma

If $M_1; S_1; D_1 \implies M_2; S_2; D_2$ *is a step of the generalized matching algorithm, then* σ *is a matcher of* $M_1; S_1; D_1$ *iff it is a matcher of* $M_2; S_2; D_2$*.*

Proof. A step of the algorithm is an application of one of the rules, thus it has to hold for each individually.

- Mer: $M_1 = M_2$, $D_1 = D_2$. The rest follows from the definition of the intersection $\sqcap$.

Soundness and Completeness

If $\mathfrak{M}$ on input $t \preceq s$, λ and $\mathcal{R}$ terminates on $\emptyset; S; D$, then by induction on the length of a derivation $\{t \preceq s\}; \emptyset; 1 \Longrightarrow^{+} \emptyset; S; D$, we can conclude that if the constraints $\bigotimes_{x \approx \mathbf{r}_\delta \in S} \delta \otimes D \geq \lambda$ and $\bigvee_{(r,\alpha_r) \in \mathbf{r}_\delta} \alpha_r = \delta$ for all $(x \approx \mathbf{r}_\delta) \in S$ are satisfiable for some set of δ, then any substitution σ that satisfies $\bigotimes_{x \approx \mathbf{r}_\delta \in S} \delta \otimes D \geq \lambda$ and $\bigvee_{(r,\alpha_r) \in \mathbf{r}_\delta} (x\sigma = r) \wedge (\alpha_r = \delta)$ for all $(x \approx \mathbf{r}_\delta) \in S$ is an $(\mathcal{R}, \lambda)$-matcher of t to s.

Soundness and Completeness

If $\mathfrak{M}$ on input $t \preceq s$, λ and $\mathcal{R}$ terminates on $\emptyset; S; D$, then by induction on the length of a derivation $\{t \preceq s\}; \emptyset; 1 \Longrightarrow^{+} \emptyset; S; D$, we can conclude that if the constraints $\bigotimes_{x \approx \mathbf{r}_\delta \in S} \delta \otimes D \geq \lambda$ and $\bigvee_{(r,\alpha_r) \in \mathbf{r}_\delta} \alpha_r = \delta$ for all $(x \approx \mathbf{r}_\delta) \in S$ are satisfiable for some set of δ, then any substitution σ that satisfies $\bigotimes_{x \approx \mathbf{r}_\delta \in S} \delta \otimes D \geq \lambda$ and $\bigvee_{(r,\alpha_r) \in \mathbf{r}_\delta} (x\sigma = r) \wedge (\alpha_r = \delta)$ for all $(x \approx \mathbf{r}_\delta) \in S$ is an $(\mathcal{R}, \lambda)$-matcher of t to s.

Thus, it suffices to solve the set of constraints
$$\bigcup_{(x \approx \mathbf{r}_\delta) \in S} \{ \bigvee_{(r,\alpha_r) \in \mathbf{r}_\delta} \alpha_r = \delta\} \cup \{ \bigotimes_{x \approx \mathbf{r}_\delta \in S} \delta \otimes D \geq \lambda\}$$
and then obtaining the proximity degrees and respective classes from the equations in S.

Obtaining Solutions

Taking the example from above with output $\emptyset; \{x \approx \mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))\}; \mathcal{R}(f,g)$, constraints are now obtained by conjuncting:

- $\lambda \leq 1 \otimes \mathcal{R}(f,g) \otimes \delta_1 \otimes \delta_2$
- $$\bigvee_{(r,\alpha_r) \in \mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))} \alpha_r = \delta_1 \otimes \delta_2$$

Obtaining Solutions

Taking the example from above with output $\emptyset; \{x \approx \mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))\}; \mathcal{R}(f,g)$, constraints are now obtained by conjuncting:

- $\lambda \leq 1 \otimes \mathcal{R}(f,g) \otimes \delta_1 \otimes \delta_2$
- $\bigvee_{(r,\alpha_r) \in \mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))} \alpha_r = \delta_1 \otimes \delta_2$

If we expand this, it looks like:

$$
\begin{aligned}
&\lambda \leq \mathcal{R}(f,g) \otimes \delta_1 \otimes \delta_2 \wedge (\\
&\delta_1 \otimes \delta_2 = \mathcal{R}(g(b,c), f(a,b)) \otimes \mathcal{R}(g(b,c), h(c,d)) \vee\\
&\delta_1 \otimes \delta_2 = \mathcal{R}(h(d,a), f(a,b)) \otimes \mathcal{R}(h(d,a), h(c,d)) \vee\\
&\delta_1 \otimes \delta_2 = \mathcal{R}(f(b,d), f(a,b)) \otimes \mathcal{R}(f(b,d), h(c,d)) \vee\\
&\ldots\\
&)
\end{aligned}
$$

Obtaining Solutions

Taking the example from above with output $\emptyset; \{x \approx \mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))\}; \mathcal{R}(f,g)$, constraints are now obtained by conjuncting:

- $\lambda \leq 1 \otimes \mathcal{R}(f,g) \otimes \delta_1 \otimes \delta_2$
- $\bigvee_{(r,\alpha_r) \in \mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))} \alpha_r = \delta_1 \otimes \delta_2$

If we expand this, it looks like:

$$\lambda \leq \mathcal{R}(f,g) \otimes \delta_1 \otimes \delta_2 \wedge ($$
$$\delta_1 \otimes \delta_2 = \mathcal{R}(g(b,c), f(a,b)) \otimes \mathcal{R}(g(b,c), h(c,d)) \vee$$
$$\delta_1 \otimes \delta_2 = \mathcal{R}(h(d,a), f(a,b)) \otimes \mathcal{R}(h(d,a), h(c,d)) \vee$$
$$\delta_1 \otimes \delta_2 = \mathcal{R}(f(b,d), f(a,b)) \otimes \mathcal{R}(f(b,d), h(c,d)) \vee$$
$$\dots$$
$$)$$

If we had more variables, we would have more clauses.

Obtaining Solutions

With our values $\lambda = 0.3$ and $\mathcal{R} :=$

- $\{f \approx_{0.9} g, g \approx_{0.8} h, h \approx_{0.25} f\} \cup$
- $\{a \approx_{0.95} b, b \approx_{0.75} c, c \approx_{0.85} d, d \approx_{0.82} a\}$,

plugged in, we get:

$$0.3 \leq 0.9 \otimes \delta_1 \otimes \delta_2 \wedge ($$

$$\delta_1 \otimes \delta_2 = 0.9 \otimes 0.95 \otimes 0.75 \otimes 0.8 \otimes 0.75 \otimes 0.85 \vee$$

$$\delta_1 \otimes \delta_2 = 0.25 \otimes 0.82 \otimes 0.95 \otimes 0.8 \otimes 0.85 \otimes 0.82 \vee$$

$$\delta_1 \otimes \delta_2 = 1 \otimes 0.95 \otimes 0 \otimes 0.25 \otimes 0.75 \otimes 1 \vee$$

$$\ldots$$

$$)$$

Compact representation

How does an expression like $\mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))$ look like?

Compact representation

How does an expression like $\mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))$ look like?

First the individual proximity classes:
$\mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) =$
$\{(f(a,b),1),(g(a,b),0.9),\ldots,(g(b,c),0.9 \otimes 0.95 \otimes 0.75),$
$\ldots,(h(c,d),0.25 \otimes 0 \otimes 0)\}$

in compact representation: $\{\{(f,1),(g,0.9),(h,0.25)\}$
$(\{(a,1),(b,0.95),(c,0),(d,0.82)\},\{(a,0.95),(b,1),(c,0.75),(d,0)\})\}$

Compact representation

How does an expression like $\mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))$ look like?

$\mathbf{pc}_{\mathcal{R},\delta_1}(h(c,d)) =$
$\{(h(c,d),1),(g(c,d),0.8),\ldots,g(b,c),0.8 \otimes 0.75 \otimes 0.85),$
$\ldots,(h(c,d),1 \otimes 0 \otimes 0)\}$

in compact representation: $\{\{(f,0.25),(g,0.8),(h,1)\}$
$(\{(a,0),(b,0.75),(c,1),(d,0.85)\},\{(a,0.82),(b,0),(c,0.85),(d,1)\})\}$

Compact representation

How does an expression like $\mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))$ look like?

$$\{\mathbf{pc}_{\mathcal{R},\delta_1}(f(a,b)) \sqcap \mathbf{pc}_{\mathcal{R},\delta_2}(h(c,d))\} =$$
$$\{\{(f, 0.25 \otimes 1), (g, 0.9 \otimes 0.8), (h, 0.25 \otimes 1)\}(\ldots)\}$$

Compact Representation: Degree constraints

Degree constraints

If a variable x has to be matched to n different terms $t_1, \ldots, t_n$ with $|Pos(t_i)| = k$, then without a compact representation, then for every variable, we get an exponential blowup in k of the number of disjunctions. If we use compact representation, this gets reduced to around k disjunctions nested in conjunctions. The exact number depends on the cardinality of $\mathcal{R}$ in each arity.

Representation: Correctness

If we use compact representation, then we have to reformulate $x \approx \mathbf{u}$ to $x \approx \tau(\mathbf{u})$ where $\tau(\mathbf{u}) = \{(t, \alpha) \in \mathcal{T} \times [0, 1] \mid Pos(\mathbf{u}) = Pos(t) \wedge \forall_{p \in Pos(\mathbf{u})} (\exists_{(s,\beta) \in \mathbf{u}|_p} s = t|_p \wedge \beta = \alpha)\}$, as in, the set of terms spanned by the compact representation.