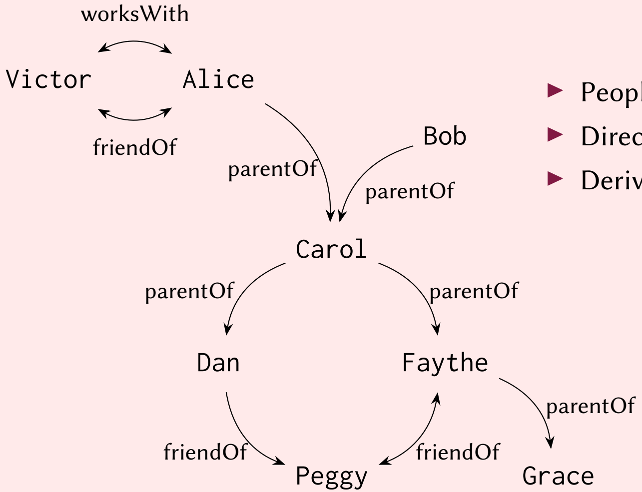
Explaining Results of Path Queries on Graphs: Single-Path Results for Context-Free Path Queries

Jelle Hellings

Department of Computer Science,
University of California, Davis,
Davis, CA 95616-8562, USA

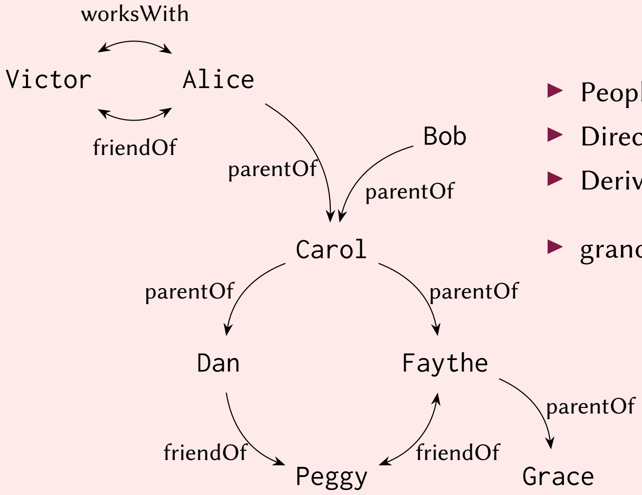
UCDAVIS
UNIVERSITY OF CALIFORNIA

Edge-labeled graphs and queries



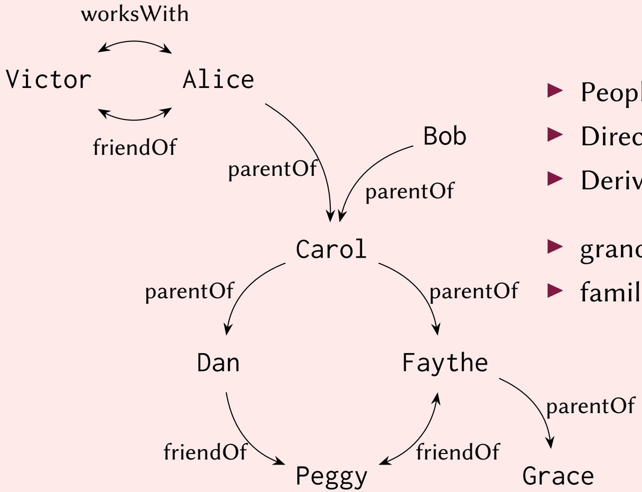
- People are *nodes*.
- Direct relationships are *edges*.
- Derivable relationships are *queries*.

Edge-labeled graphs and queries



- People are *nodes*.
- Direct relationships are *edges*.
- Derivable relationships are *queries*.
- grandParentOf := parentOf ∘ parentOf.

Edge-labeled graphs and queries



- People are *nodes*.
- Direct relationships are *edges*.
- Derivable relationships are *queries*.

- grandParentOf := parentOf $\circ$ parentOf.
- familyOf := (parentOf $\cup$ childOf)*.

Path queries: Expressing queries via formal languages

- Simple queries represent graph navigation via a path.
- Capture this navigation via the path labeling.
- Express the labeling of interest via a formal language
  E.g., regular languages or context-free languages.

Path queries: Expressing queries via formal languages

- Simple queries represent graph navigation via a path.
- Capture this navigation via the path labeling.
- Express the labeling of interest via a formal language
  E.g., regular languages or context-free languages.

This work: Context-free path queries

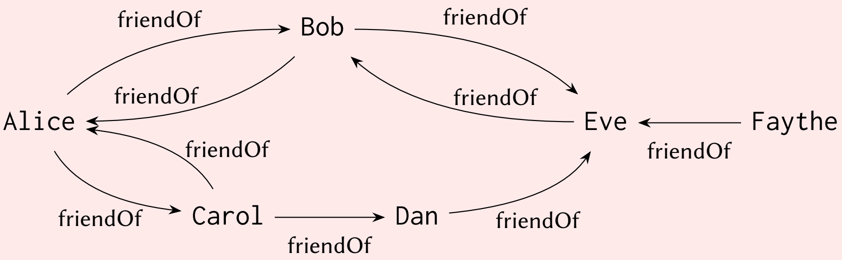
A grammar $\mathscr{C} = (\mathcal{N}, \Sigma, \mathcal{P})$ is

- a set of non-terminals $\mathcal{N}$;
- a set of alphabet symbols Σ; and
- a set of production rules $\mathcal{P}$ of the form $\text{A} \mapsto \sigma$ or $\text{A} \mapsto \text{B C}$.

Example: The context-free grammar for indirectFriendOf := friendOf$^+$

$\mathcal{N} = \{\text{A}\}$, $\Sigma = \{\text{friendOf}\}$, and $\mathcal{P} = \{\text{A} \rightarrow \text{friendOf}, \text{A} \rightarrow \text{A A}\}$.

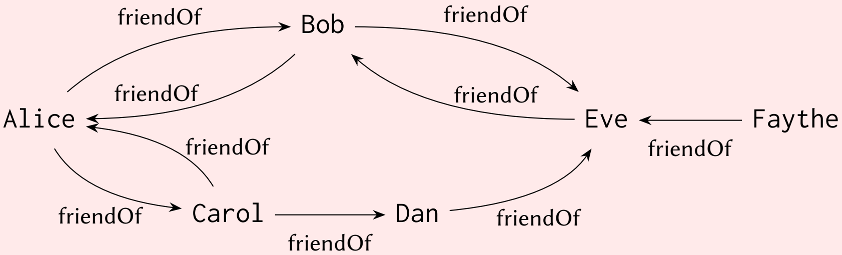
Limitations of traditional path query evaluation



Problem: Alice wants to contact Eve via friends

indirectFriendOf

Limitations of traditional path query evaluation

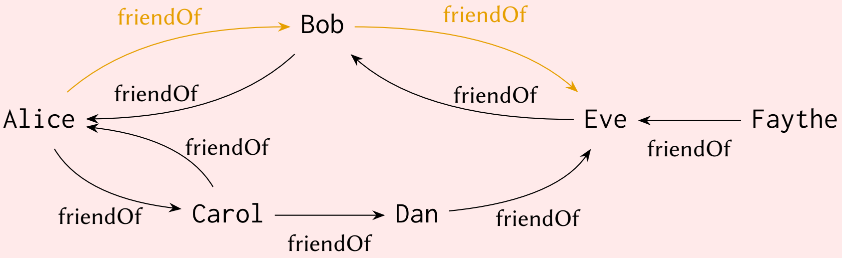


Problem: Alice wants to contact Eve via friends

indirectFriendOf —evaluates to→

Alice	Alice
Alice	Carol
...	...
Alice	Eve
...	...

Limitations of traditional path query evaluation

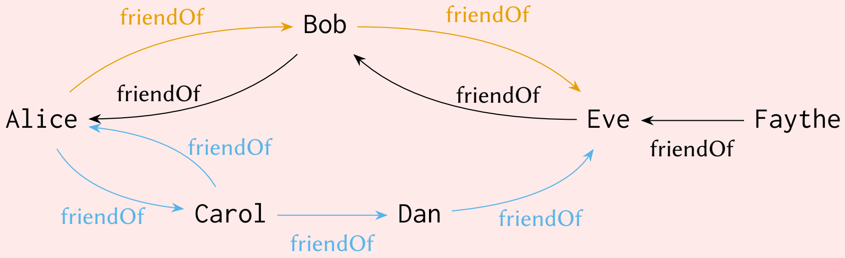


Problem: Alice wants to contact Eve via friends

indirectFriendOf —evaluates to→

Alice	Alice
Alice	Carol
...	...
Alice	Eve
...	...

Limitations of traditional path query evaluation



Problem: Alice wants to contact Eve via friends

indirectFriendOf —evaluates to→

Alice	Alice
Alice	Carol
...	...
Alice	Eve
...	...

The single-path semantics

The evaluation $\text{single}(q|_{\mathfrak{G}})$ of *path query* q specified by *language* $\mathcal{L}$ on *graph* $\mathfrak{G}$ yields

$$\text{single}(q|_{\mathfrak{G}}) = \{m\pi n \mid \pi \text{ is a shortest path in } \mathfrak{G} \text{ such that } \text{trace}(\pi) \in \mathcal{L}\}.$$

indirectFriendOf

The single-path semantics

The evaluation $\mathrm{single}(q|_{\mathfrak{G}})$ of *path query q* specified by *language* $\mathcal{L}$ on *graph* $\mathfrak{G}$ yields

$$\mathrm{single}(q|_{\mathfrak{G}}) = \{m\pi n \mid \pi \text{ is a shortest path in } \mathfrak{G} \text{ such that } \mathrm{trace}(\pi) \in \mathcal{L}\}.$$

indirectFriendOf $\xrightarrow[\text{single-path}]{\text{evaluates to}}$

Alice friendOf Bob friendOf Alice
Alice friendOf Carol
...
Alice friendOf Bob friendOf Eve
...

Representing the paths of interest

- Edge-labeled graphs are *finite automata*.
- The traces of paths from one node to another represent a *regular language*.

Representing the paths of interest

- Edge-labeled graphs are *finite automata*.
- The traces of paths from one node to another represent a *regular language*.

Lemma (Bar-Hillel et al.)

Let $\mathscr{C} = (\mathcal{N}, \Sigma, \mathcal{P})$ *be a grammar, let* $\mathfrak{G} = (\mathcal{V}, \Sigma, \delta)$ *be a graph, let* $\textsc{a} \in \mathcal{N}$*, and let* $m, n \in \mathcal{V}$*. The language* $\mathcal{L}(\mathscr{C}; \textsc{a}) \cap \mathcal{L}(\mathfrak{G}; m, n)$ *can be represented by a grammar.*

Representing the paths of interest

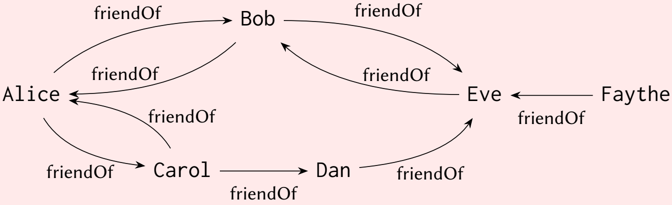
- Edge-labeled graphs are *finite automata*.
- The traces of paths from one node to another represent a *regular language*.

Lemma (Bar-Hillel et al.)

Let $\mathscr{C} = (\mathcal{N}, \Sigma, \mathcal{P})$ *be a grammar, let* $\mathfrak{G} = (\mathcal{V}, \Sigma, \delta)$ *be a graph, let* $\textsc{a} \in \mathcal{N}$*, and let* $m, n \in \mathcal{V}$*. The language* $\mathcal{L}(\mathscr{C}; \textsc{a}) \cap \mathcal{L}(\mathfrak{G}; m, n)$ *can be represented by a grammar.*

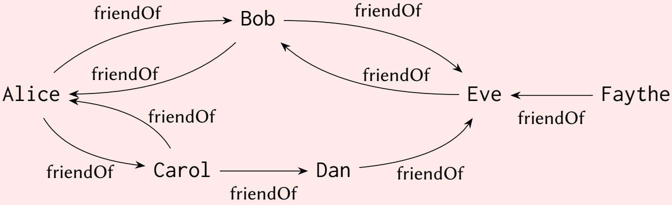
- Mismatch: many paths have the same trace!
- Solution: combine encoding of grammar and graph via *annotated grammar*.

Annotated grammar: Example



$$\text{indirectFriendOf} := \{\text{A} \longrightarrow \text{friendOf}, \text{A} \longrightarrow \text{A A}\}.$$

Annotated grammar: Example

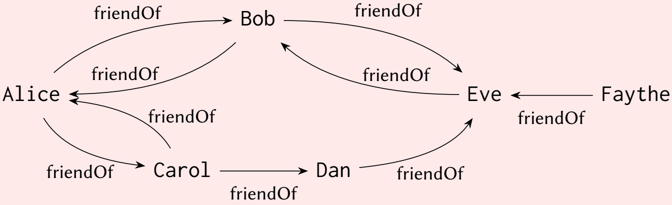


$$\mathrm{indirectFriendOf} := \{\textsc{a} \longrightarrow \mathrm{friendOf}, \textsc{a} \longrightarrow \textsc{a}\,\textsc{a}\}.$$

Annotated grammar $\mathscr{C}|_{\mathfrak{G}} = (\mathcal{N}|_{\mathfrak{G}}, \Sigma, \mathcal{P}|_{\mathfrak{G}})$ with

- $\mathcal{N}|_{\mathfrak{G}} = \{\mathsf{A}|_{mn} \mid m, n \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}\} \cup \{\mathsf{A}|_{\mathsf{F}n} \mid n \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}\}$; and
- $\mathcal{P}|_{\mathfrak{G}} = P_\Sigma \cup P_{\mathcal{N}}$ with
  - $P_\Sigma = \{\mathsf{A}|_{mn} \mapsto \sigma \mid (m, \sigma, n) \in \delta \wedge (\textsc{a} \mapsto \sigma) \in \mathcal{P}\}$; and
  - $P_{\mathcal{N}} = \{\mathsf{A}|_{mn} \mapsto \textsc{b}|_{mo}\, \textsc{c}|_{on} \mid (\textsc{a} \mapsto \textsc{b}\,\textsc{c}) \in \mathcal{P}\}$.

Annotated grammar: Example



$$\text{indirectFriendOf} := \{\mathrm{A} \longrightarrow \text{friendOf}, \mathrm{A} \longrightarrow \mathrm{A}\ \mathrm{A}\}.$$

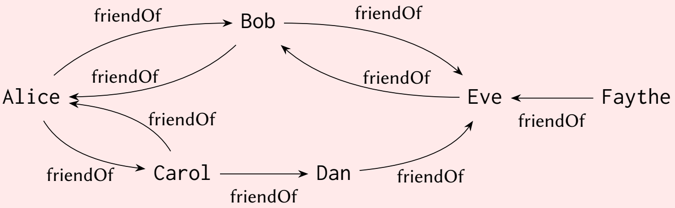
Annotated grammar $\mathscr{C}|_{\mathfrak{G}} = (\mathcal{N}|_{\mathfrak{G}}, \Sigma, \mathcal{P}|_{\mathfrak{G}})$ with

- $\mathcal{N}|_{\mathfrak{G}} = \{\mathrm{A}|_{mn} \mid m, n \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}\}\} \cup \{\mathrm{A}|_{\mathrm{F}n} \mid n \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}\}\}$; and
- $\mathcal{P}|_{\mathfrak{G}} = P_{\Sigma} \cup P_{\mathcal{N}}$ with
  - $P_{\Sigma} = \{\mathrm{A}|_{mn} \mapsto \sigma \mid (m, \sigma, n) \in \delta \wedge (\mathrm{A} \mapsto \sigma) \in \mathcal{P}\}$; and
  - $P_{\mathcal{N}} = \{\mathrm{A}|_{mn} \mapsto \mathrm{B}|_{mo}\ \mathrm{C}|_{on} \mid (\mathrm{A} \mapsto \mathrm{B}\ \mathrm{C}) \in \mathcal{P}\}$.

Deriving a path from Alice to Eve

$$\mathrm{A}|_{\texttt{AliceEve}}$$

Annotated grammar: Example



$$\mathsf{indirectFriendOf} := \{\textsc{a} \rightarrow \mathsf{friendOf}, \textsc{a} \rightarrow \textsc{a}\,\textsc{a}\}.$$

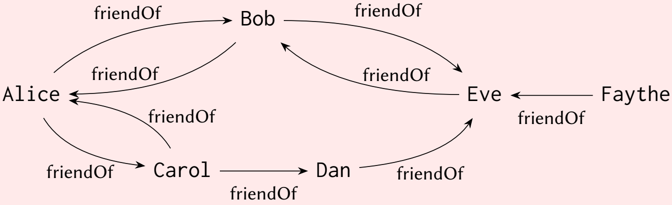
Annotated grammar $\mathscr{C}|_{\mathfrak{G}} = (\mathcal{N}|_{\mathfrak{G}}, \Sigma, \mathcal{P}|_{\mathfrak{G}})$ with

- $\mathcal{N}|_{\mathfrak{G}} = \{\mathsf{A}|_{mn} \mid m, n \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}\} \cup \{\mathsf{A}|_{\mathsf{F}n} \mid n \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}\}$; and
- $\mathcal{P}|_{\mathfrak{G}} = P_{\Sigma} \cup P_{\mathcal{N}}$ with
  - $P_{\Sigma} = \{\mathsf{A}|_{mn} \mapsto \sigma \mid (m, \sigma, n) \in \delta \wedge (\textsc{a} \mapsto \sigma) \in \mathcal{P}\}$; and
  - $P_{\mathcal{N}} = \{\mathsf{A}|_{mn} \mapsto \textsc{b}|_{mo}\ \textsc{c}|_{on} \mid (\textsc{a} \mapsto \textsc{b}\ \textsc{c}) \in \mathcal{P}\}$.

Deriving a path from Alice to Eve

$$\mathsf{A}|_{\mathtt{AliceCarol}}\ \mathsf{A}|_{\mathtt{CarolEve}}$$

Annotated grammar: Example



$$\text{indirectFriendOf} := \{\textsc{a} \longrightarrow \text{friendOf}, \textsc{a} \longrightarrow \textsc{a}\ \textsc{a}\}.$$

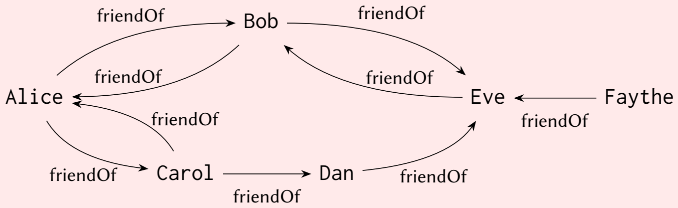
Annotated grammar $\mathscr{C}|_{\mathfrak{G}} = (\mathcal{N}|_{\mathfrak{G}}, \Sigma, \mathcal{P}|_{\mathfrak{G}})$ with

- $\mathcal{N}|_{\mathfrak{G}} = \{\mathsf{A}|_{mn} \mid m, n \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}\} \cup \{\mathsf{A}|_{\mathsf{F}n} \mid n \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}\}$; and
- $\mathcal{P}|_{\mathfrak{G}} = P_\Sigma \cup P_\mathcal{N}$ with
  - $P_\Sigma = \{\mathsf{A}|_{mn} \mapsto \sigma \mid (m, \sigma, n) \in \delta \wedge (\textsc{a} \mapsto \sigma) \in \mathcal{P}\}$; and
  - $P_\mathcal{N} = \{\mathsf{A}|_{mn} \mapsto \textsc{b}|_{mo}\ \textsc{c}|_{on} \mid (\textsc{a} \mapsto \textsc{b}\ \textsc{c}) \in \mathcal{P}\}$.

Deriving a path from Alice to Eve

$$\mathtt{A}|_{\mathtt{AliceCarol}}\ \mathtt{A}|_{\mathtt{CarolDan}}\ \mathtt{A}|_{\mathtt{DanEve}}$$

Annotated grammar: Example



indirectFriendOf :=

$\{\textsc{a} \longrightarrow \text{friendOf}, \textsc{a} \longrightarrow \textsc{a}\,\textsc{a}\}.$

Annotated grammar $\mathscr{C}|_{\mathfrak{G}} = (\mathcal{N}|_{\mathfrak{G}}, \Sigma, \mathcal{P}|_{\mathfrak{G}})$ with

- $\mathcal{N}|_{\mathfrak{G}} = \{\textsc{a}|_{mn} \mid m, n \in \{A, B, C, D, E\}\} \cup \{\textsc{a}|_{Fn} \mid n \in \{A, B, C, D, E\}\}$; and
- $\mathcal{P}|_{\mathfrak{G}} = P_\Sigma \cup P_\mathcal{N}$ with
  - $P_\Sigma = \{\textsc{a}|_{mn} \mapsto \sigma \mid (m, \sigma, n) \in \delta \wedge (\textsc{a} \mapsto \sigma) \in \mathcal{P}\}$; and
  - $P_\mathcal{N} = \{\textsc{a}|_{mn} \mapsto \textsc{b}|_{mo}\ \textsc{c}|_{on} \mid (\textsc{a} \mapsto \textsc{b}\ \textsc{c}) \in \mathcal{P}\}$.

Deriving a path from Alice to Eve

Alice friendOf Carol friendOf Dan friendOf Eve

Shortest string in a grammar

Algorithm MinimizeSet($\mathscr{C} = (\mathcal{N}, \Sigma, \mathcal{P})$):

$\mathcal{P}'$, *cost* := empty mapping, empty mapping.
new is a min-priority queue.
for all $(\text{A} \mapsto \sigma) \in \mathcal{P}$ **do**
    if A $\notin$ *cost* **then**
        cost[A], $\mathcal{P}'$[A] := 1, $(\text{A} \mapsto \sigma)$.
        add A to *new* with priority 1.
while *new* $\neq \emptyset$ **do**
    Take A with minimum priority in *new*.
    Remove A from *new*.
    for all $(\text{C} \mapsto \text{A B}) \in \mathcal{P}$ with B $\in$ *cost* **do**
        produce(C $\mapsto$ A B).
    for all $(\text{C} \mapsto \text{B A}) \in \mathcal{P}$ with B $\in$ *cost* **do**
        produce(C $\mapsto$ B A).
return $\{\mathcal{P}'[\text{A}] \mid \text{A} \in \mathcal{P}'\}$.

Algorithm produce(D $\mapsto$ E F):

if D $\notin$ *cost* **then**
    cost[D] := *cost*[E] + *cost*[F].
    $\mathcal{P}'$[D] := D $\mapsto$ E F.
    Add D to *new* with priority *cost*[E] + *cost*[F].
else if *cost*[D] > *cost*[E] + *cost*[F] **then**
    cost[D] := *cost*[E] + *cost*[F].
    $\mathcal{P}'$[D] := D $\mapsto$ E F.
    Lower priority of D $\in$ *new* to *cost*[E] + *cost*[F].

Theorem

MinimizeSet($\mathscr{C}$) yields a minimizing set of production rules in

$$\mathcal{O}(|\mathcal{N}|(|\mathcal{N}|\log|\mathcal{N}| + |\mathcal{P}|)).$$

Evaluating single-path semantics

$\textsc{MinimizeSetGG}(\mathscr{C} = (\mathcal{N}, \Sigma, \mathcal{P}), \mathfrak{G} = (\mathcal{V}, \Sigma, \delta))$

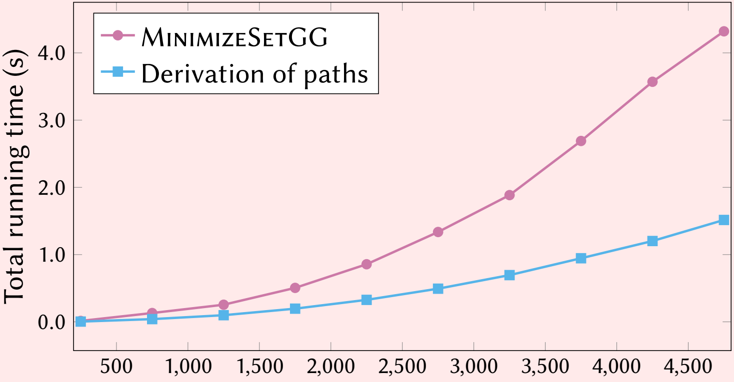
1. Use $\textsc{MinimizeSet}$ on an annotated grammar.
2. Improvement: derive annotated grammar in-place.
3. Derive shortest paths from the resulting production rules.

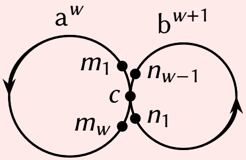
Theorem

$\textsc{MinimizeSetGG}(\mathscr{C}, \mathfrak{G})$ *yields a minimizing set of production rules in*

$$\mathcal{O}(|\mathcal{N}||\mathcal{V}|^2(|\mathcal{N}||\mathcal{V}|^2 \log(|\mathcal{N}||\mathcal{V}|^2) + |\mathcal{P}|(|\mathcal{V}|^3 + |\delta|)).$$

Cost of the single-path semantics



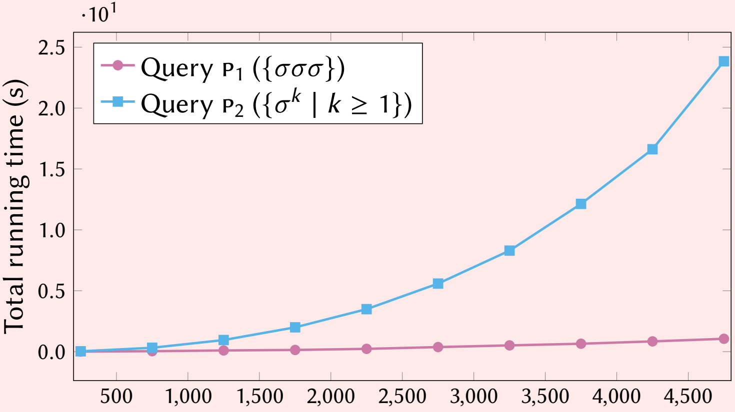


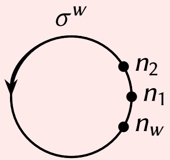
- 11, 290, 751 paths.
- Longest: 11, 286, 001 edges.
- Average: 5.640.627 edges.

$\text{Q} \mapsto \text{A Q}'$, $\quad \text{Q}' \mapsto \text{Q B}$, $\quad \text{Q} \mapsto \text{A B}$,

$\text{A} \mapsto \text{a}$, $\quad \text{B} \mapsto \text{b}$.

Grammars: Bounded vs. unbounded

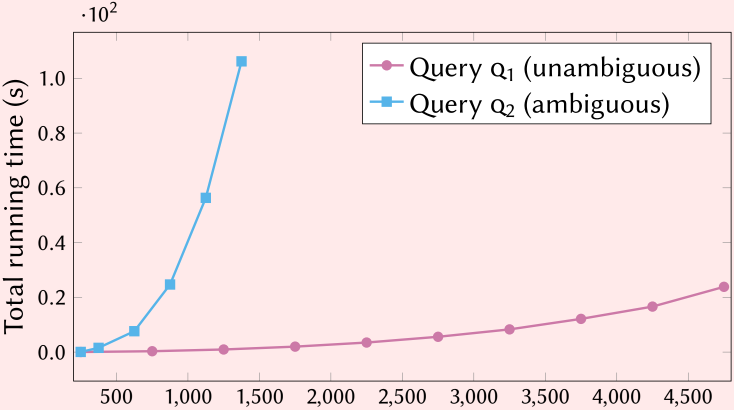


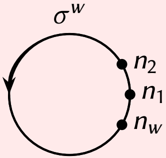


$$\mathrm{P}_1 \mapsto \mathrm{S}\ \mathrm{B} \qquad \mathrm{B} \mapsto \mathrm{S}\ \mathrm{S} \qquad \mathrm{S} \mapsto \sigma;$$

$$\mathrm{P}_2 \mapsto \mathrm{S}\ \mathrm{P}_2 \qquad \mathrm{P}_2 \mapsto \sigma \qquad \mathrm{S} \mapsto \sigma.$$

Grammars: Unambiguous vs. ambiguous





$Q_1 \mapsto S\ Q_1 \qquad Q_1 \mapsto \sigma \qquad S \mapsto \sigma;$

$Q_2 \mapsto Q_2\ Q_2 \qquad Q_2 \mapsto \sigma.$

Conclusion

Efficient answering path queries with shortest paths is possible.

Future Work

- Goal-oriented algorithms.
- High-performance and scalable algorithms.
- Optimizations for simple grammars (e.g., LL(1), LR(1)).

`https://jhellings.nl/`