Separability by Short Subsequences and Subwords

Piotr Hofman
LSV, ENS Cachan

Wim Martens
University of Bayreuth

ICDT 2015

Motivation

Motivation

What does a database theoretician do in the morning?

Motivation

What does a database theoretician do in the morning?

- Get some coffee (optional)

Motivation

What does a database theoretician do in the morning?

- Get some coffee (optional)
- Start computer

Motivation

What does a database theoretician do in the morning?

- Get some coffee (optional)
- Start computer
- Run your favorite query:

Motivation

What does a database theoretician do in the morning?

- Get some coffee (optional)
- Start computer
- Run your favorite query:

$\mathcal{Q}_{great}$ = "Who is the greatest database theoretician?"

Motivation

What does a database theoretician do in the morning?

- Get some coffee (optional)
- Start computer
- Run your favorite query:

$\mathcal{Q}_{great}$ = "Who is the greatest database theoretician?"

(It's a pretty complicated query, tweaked to your personal interests)

Motivation

$\mathcal{Q}_{great}$ = "Who is the greatest database theoretician?"

Motivation

$\mathcal{Q}_{great}$ = "Who is the greatest database theoretician?"

- One day, something strange happens

Motivation

$\mathcal{Q}_{great}$ = "Who is the greatest database theoretician?"

- One day, something strange happens
- The query returns someone you didn't expect

Motivation

$\mathcal{Q}_{great}$ = "Who is the greatest database theoretician?"

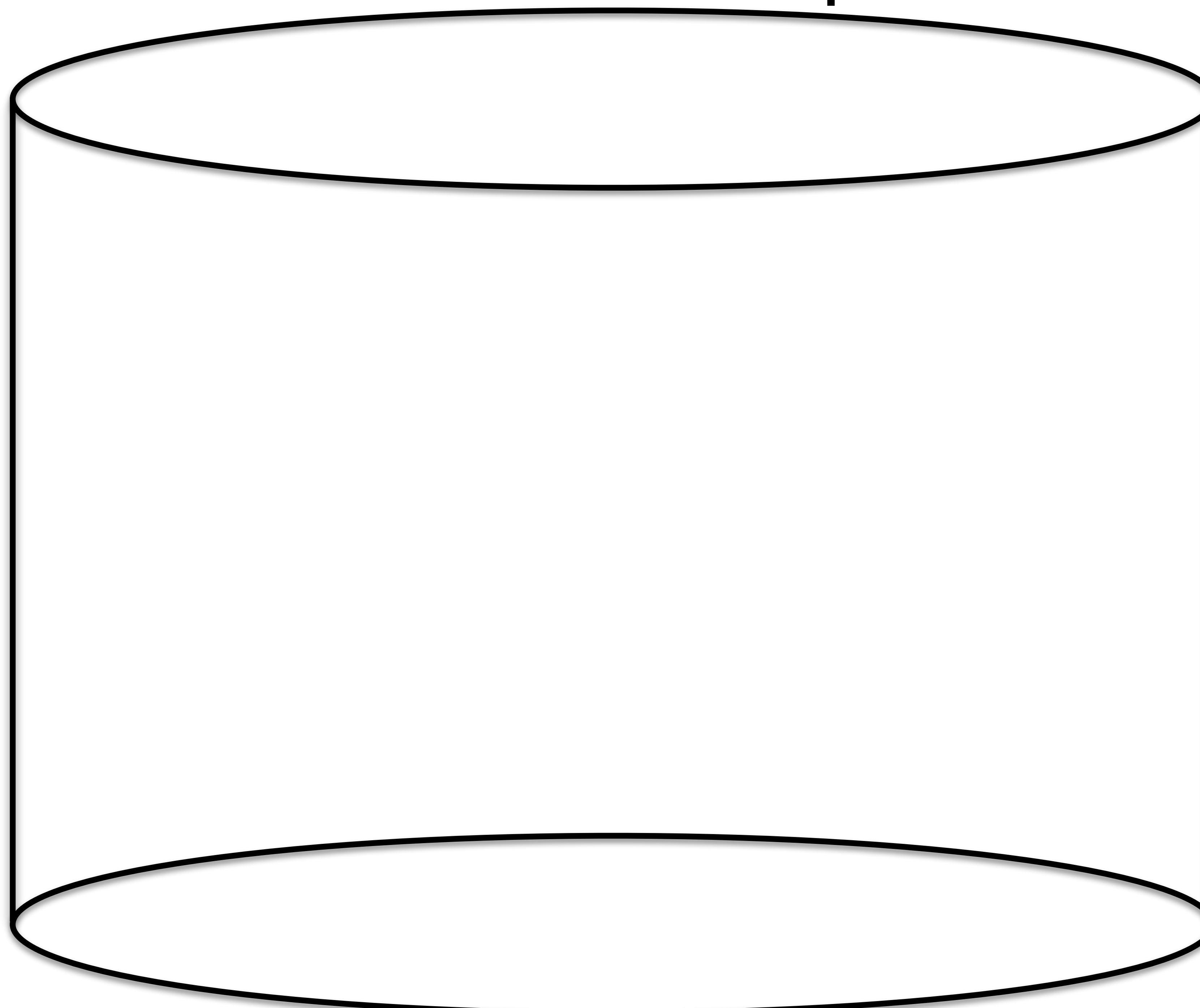
- One day, something strange happens
- The query returns someone you didn't expect
- So you wonder: "What's going on here?"

Motivation

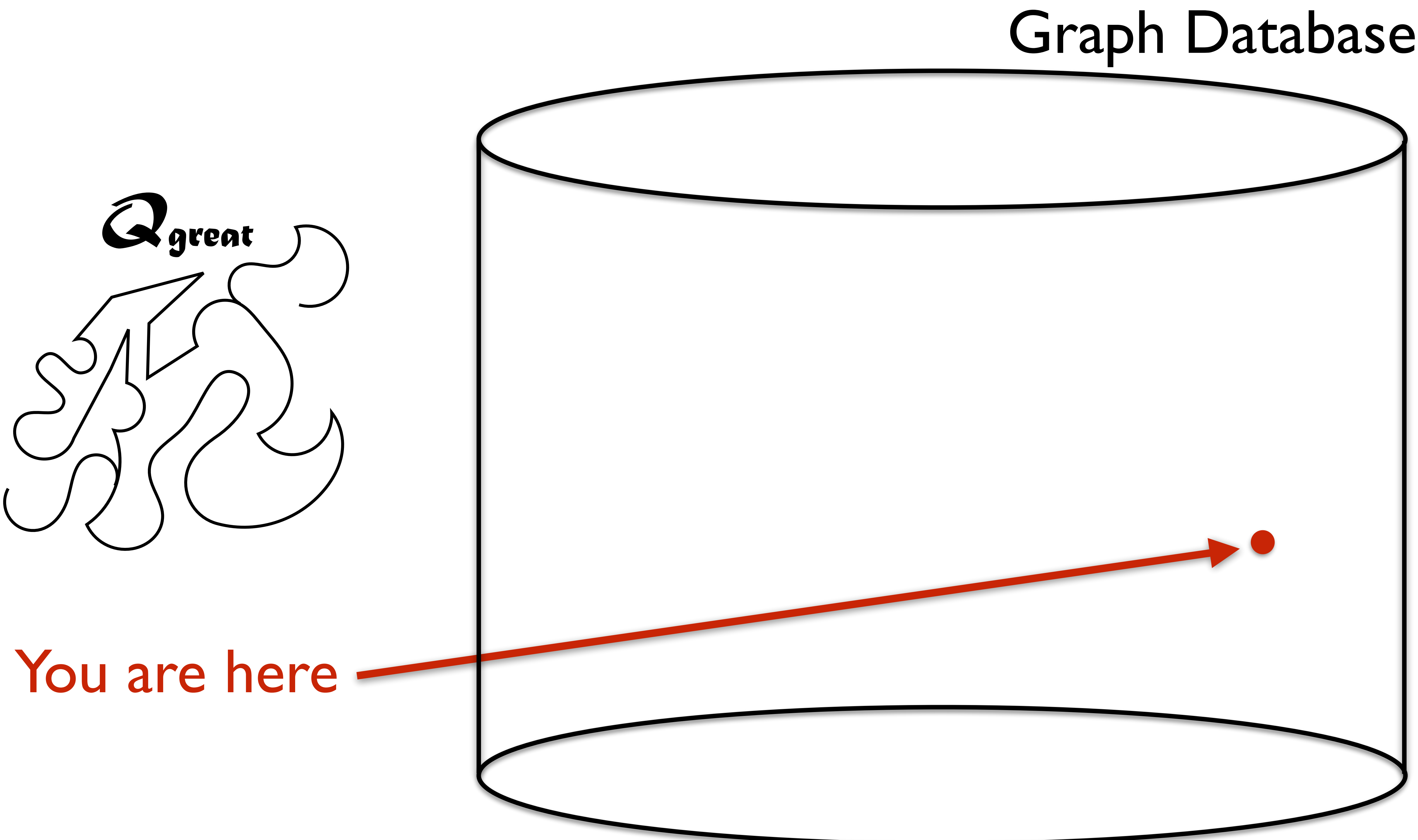
Motivation

Motivation

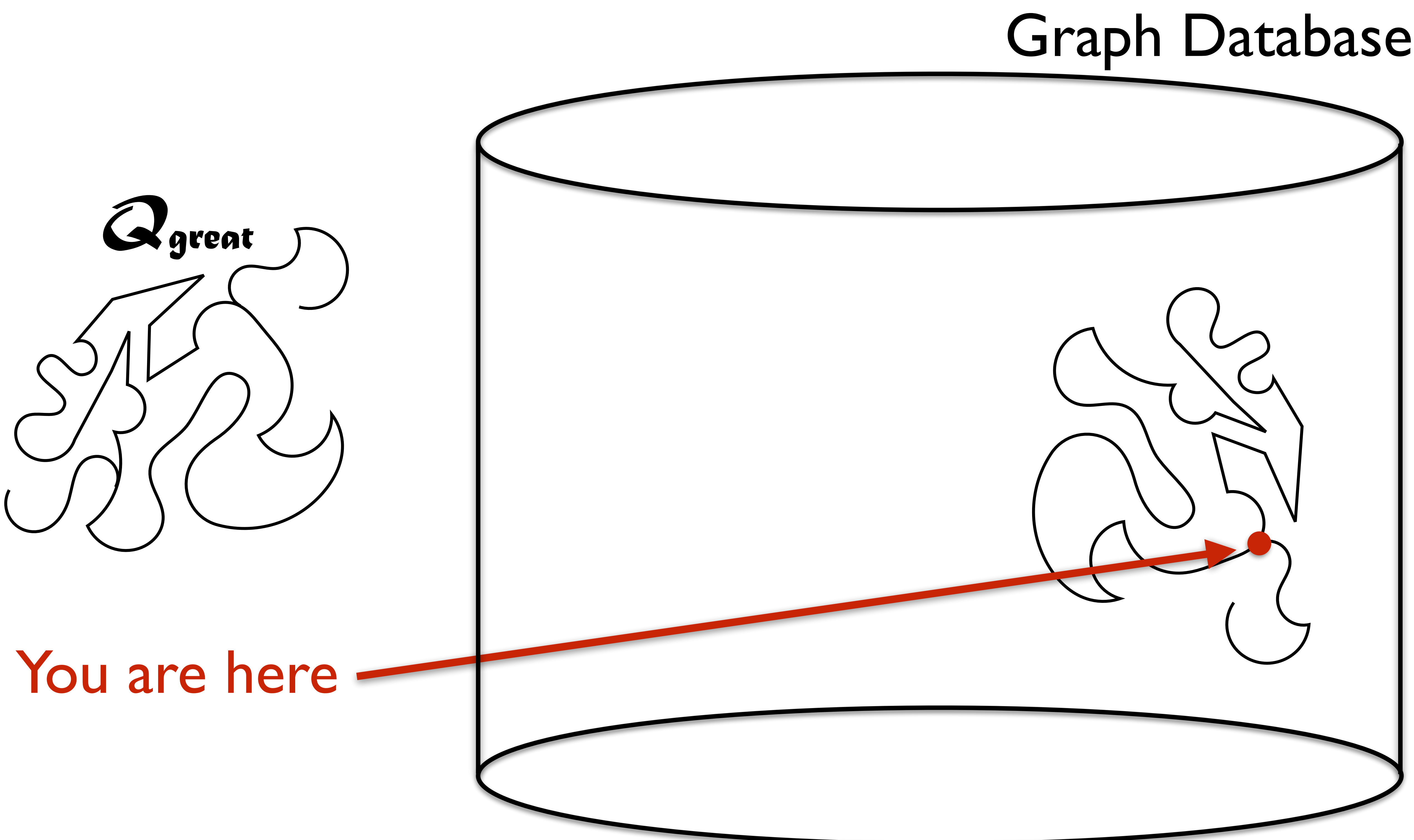
Graph Database



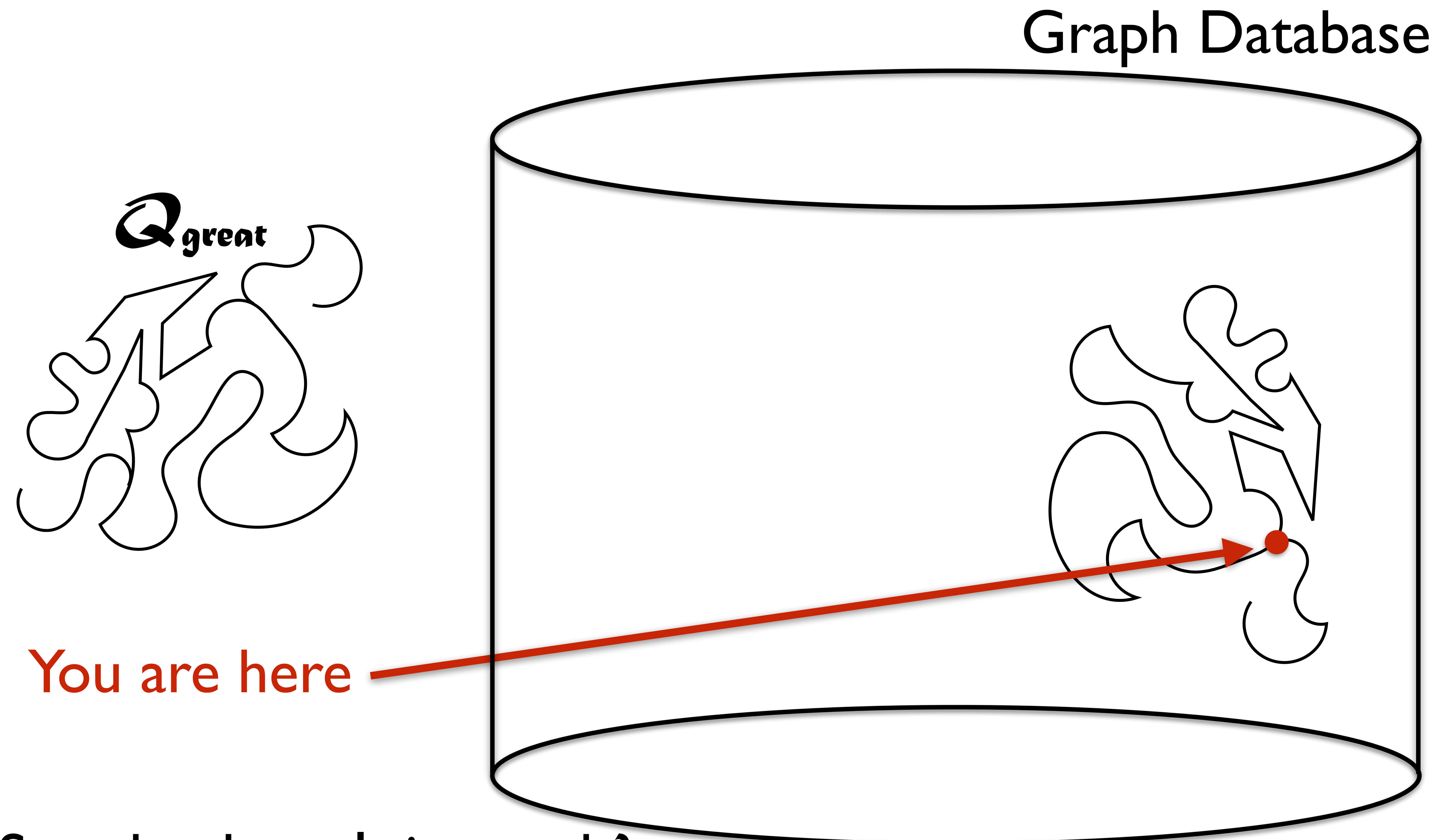
Motivation



Motivation

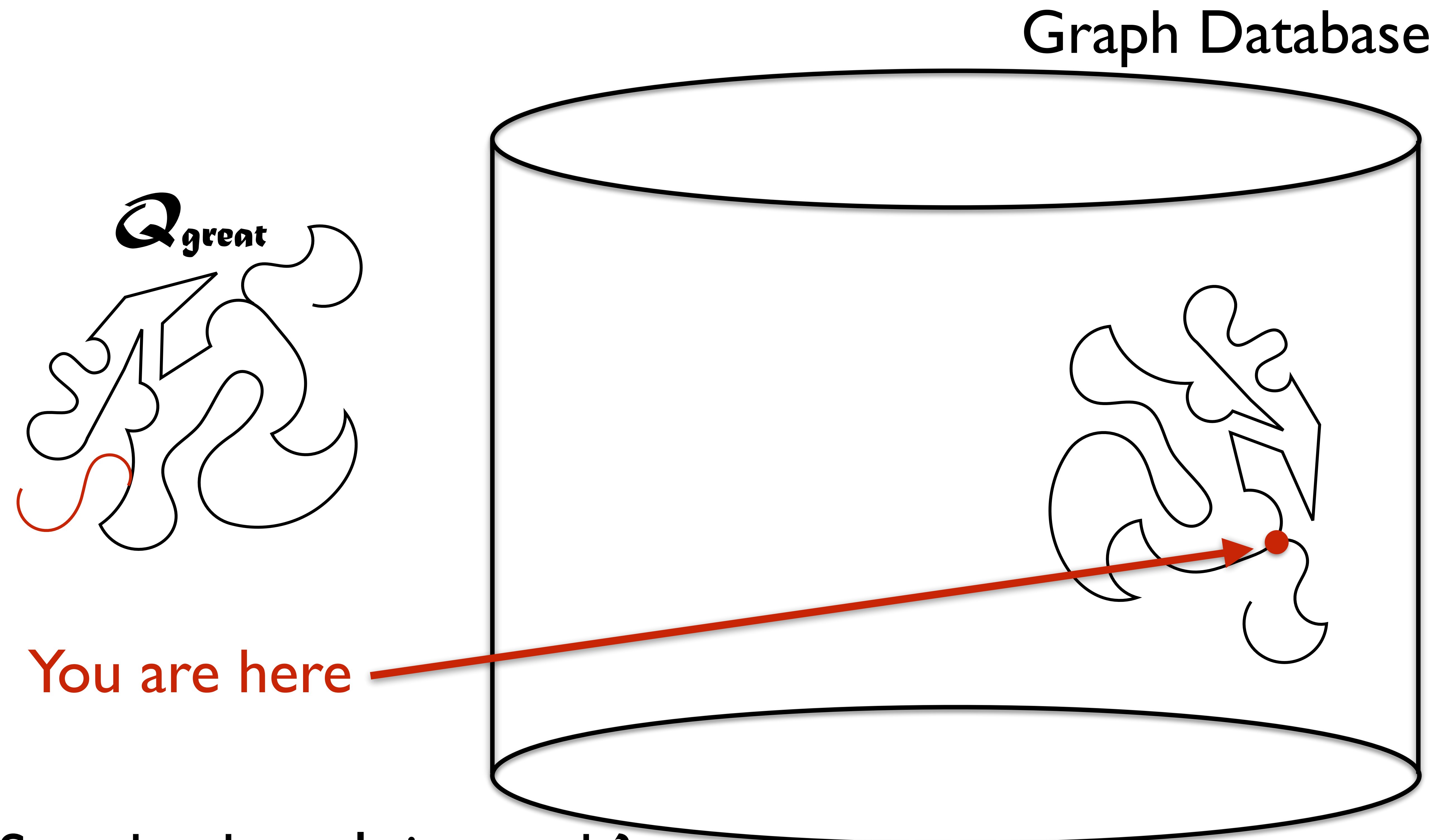


Motivation



So, why doesn't it match?

Motivation



So, why doesn't it match?
Here, there's a very simple explanation

So, we're looking for

explanations
of
why a query
doesn't return a result you expect

So, we're looking for

explanations
of
why a query
doesn't return a result you expect

How do we formalize this?

More concrete

More concrete

The data is an edge-labeled directed graph G
The query is a Regular Path Query (RPQ) r

(regular expression)

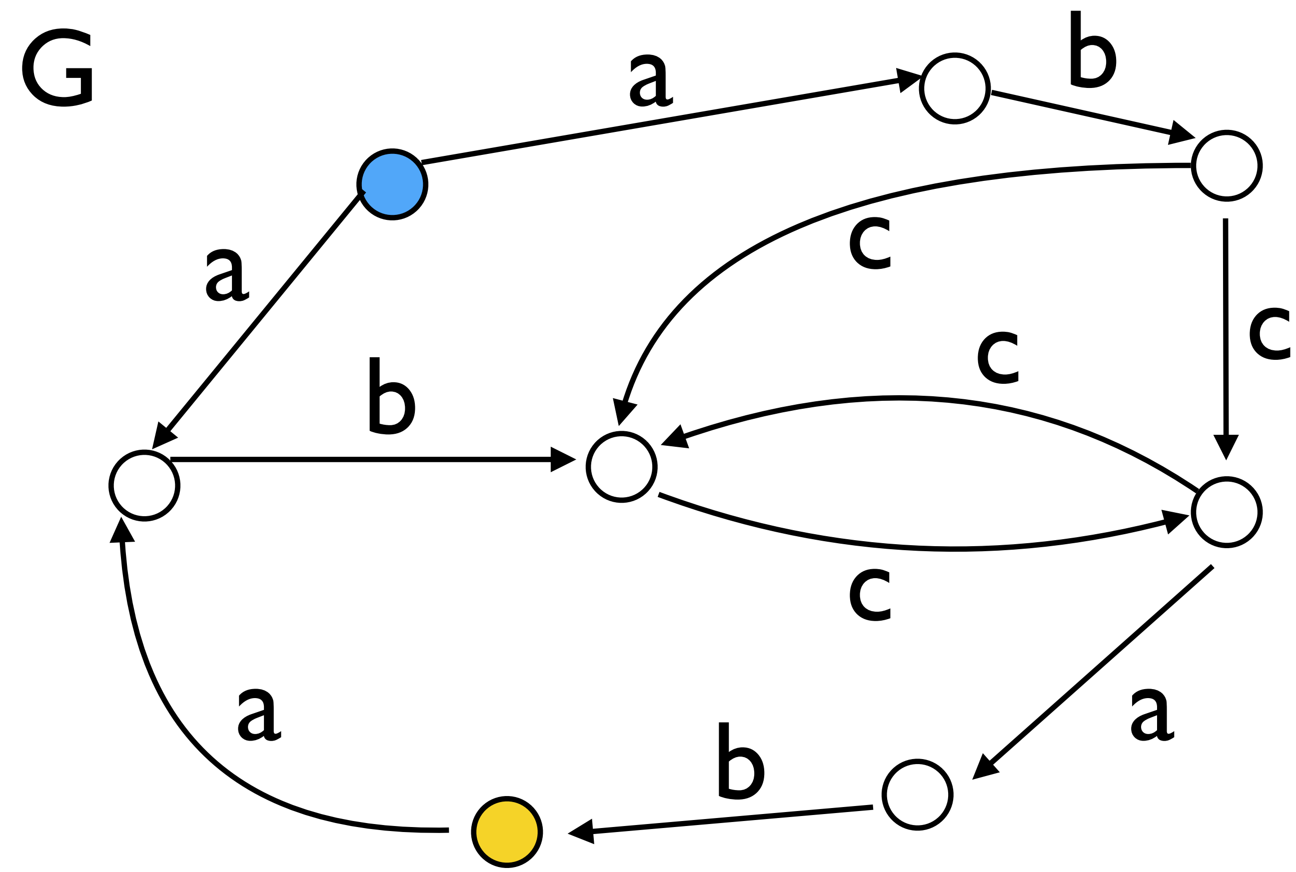
More concrete

The data is an edge-labeled directed graph G
The query is a Regular Path Query (RPQ) r

(regular expression)

An RPQ r returns
pairs of nodes (x,y)
such that there is a path from x to y in G
that is labeled by a word in L(r)

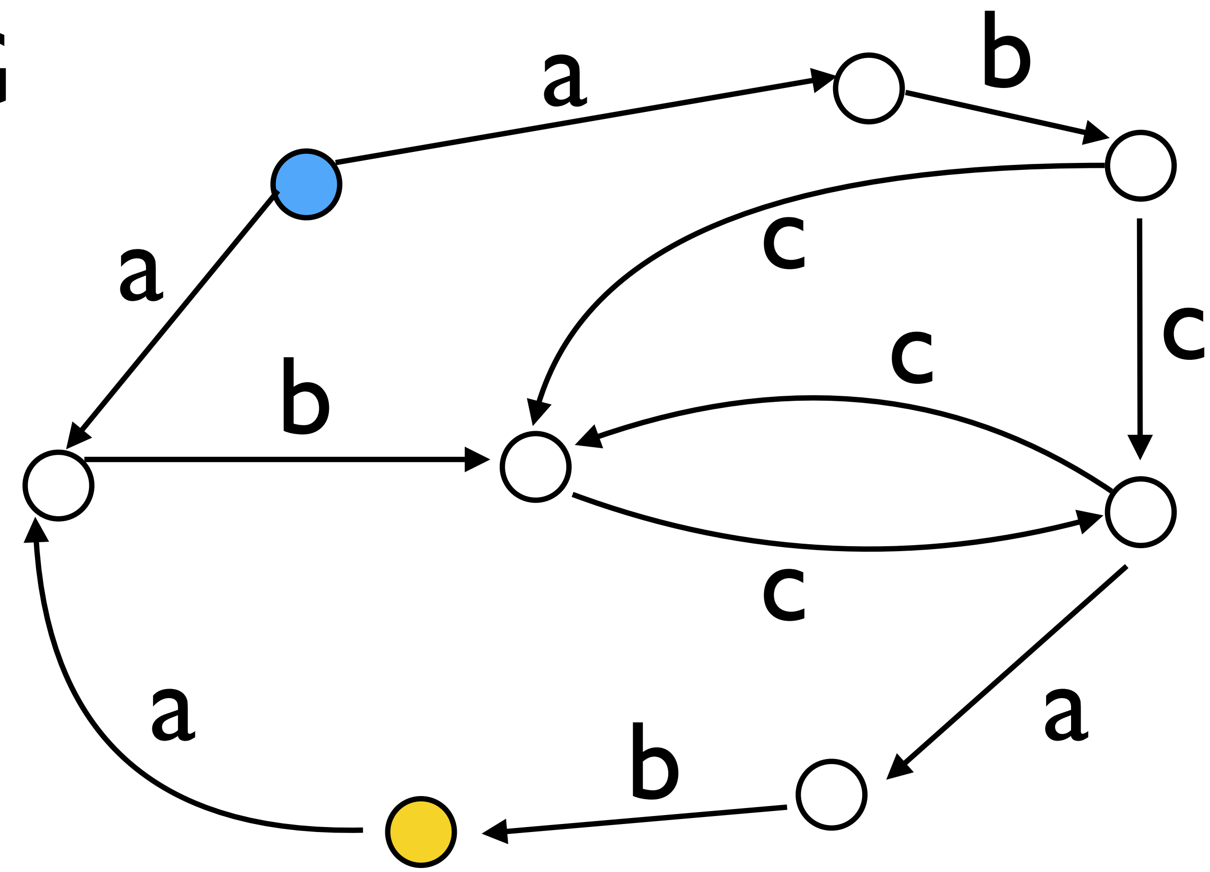
More concrete



r = ab(cc)*ab

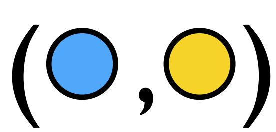
More concrete

G

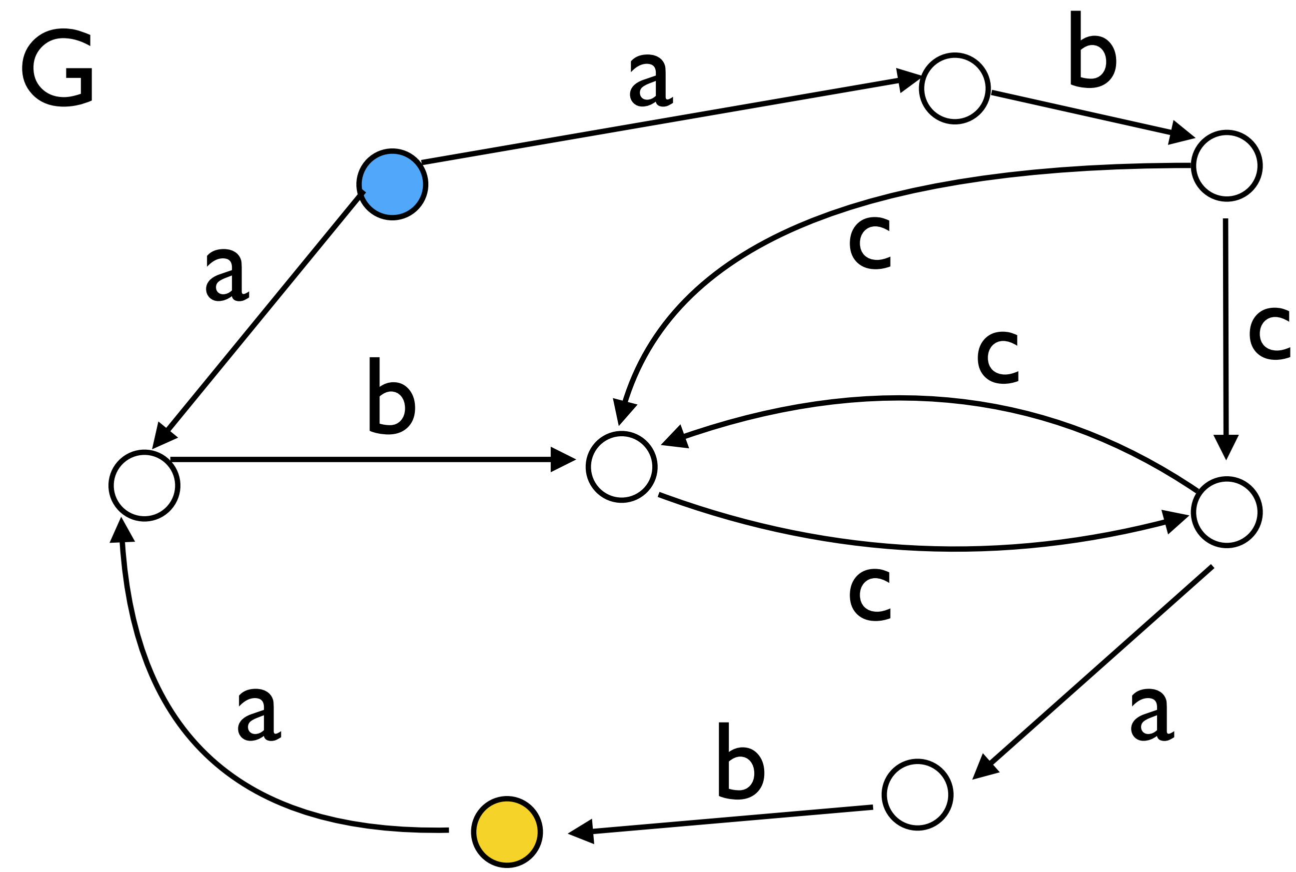


r = ab(cc)*ab

returns



More concrete



$r = ab(cc)^*ab$

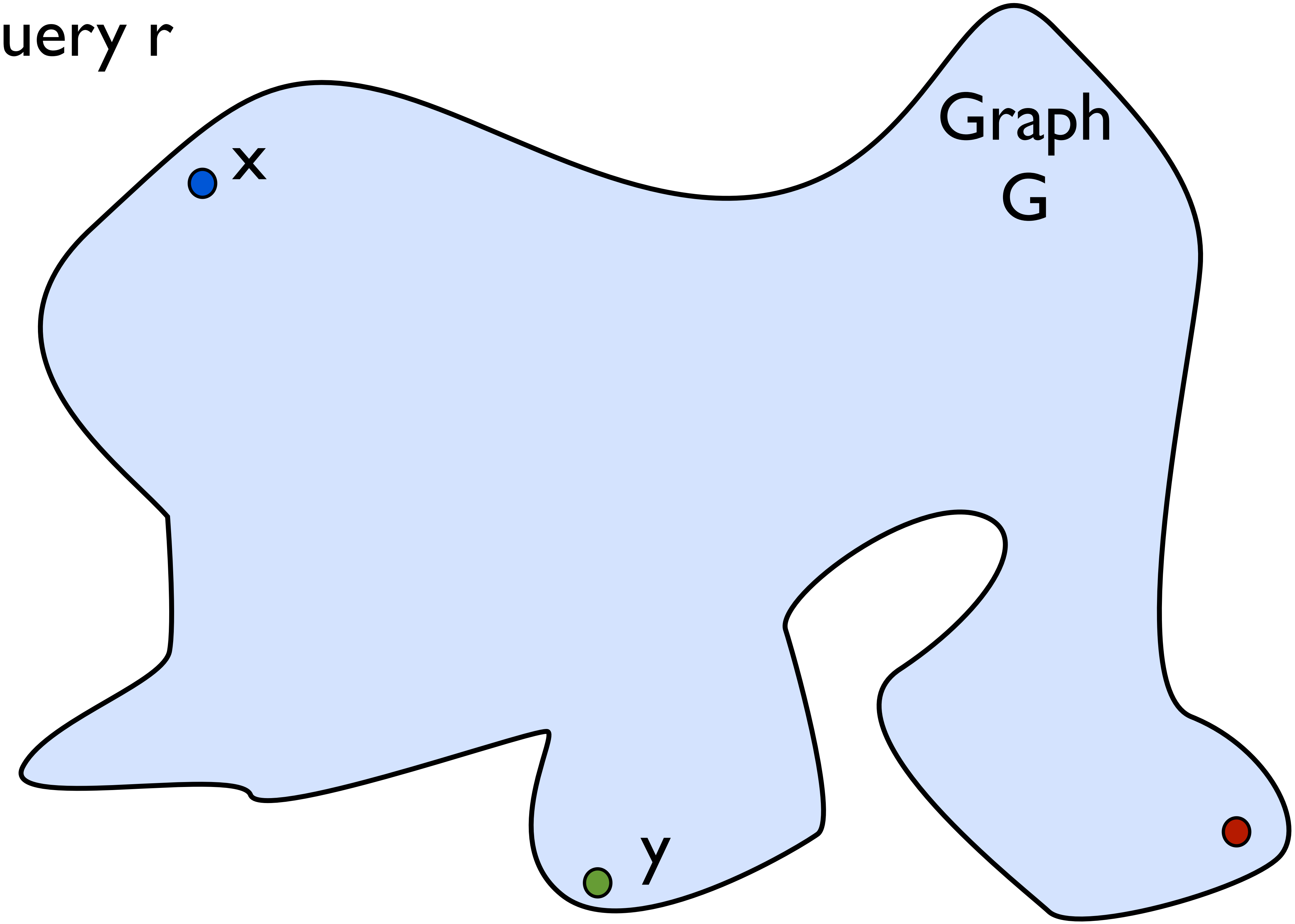
returns

(,)

but not (,)

More concrete

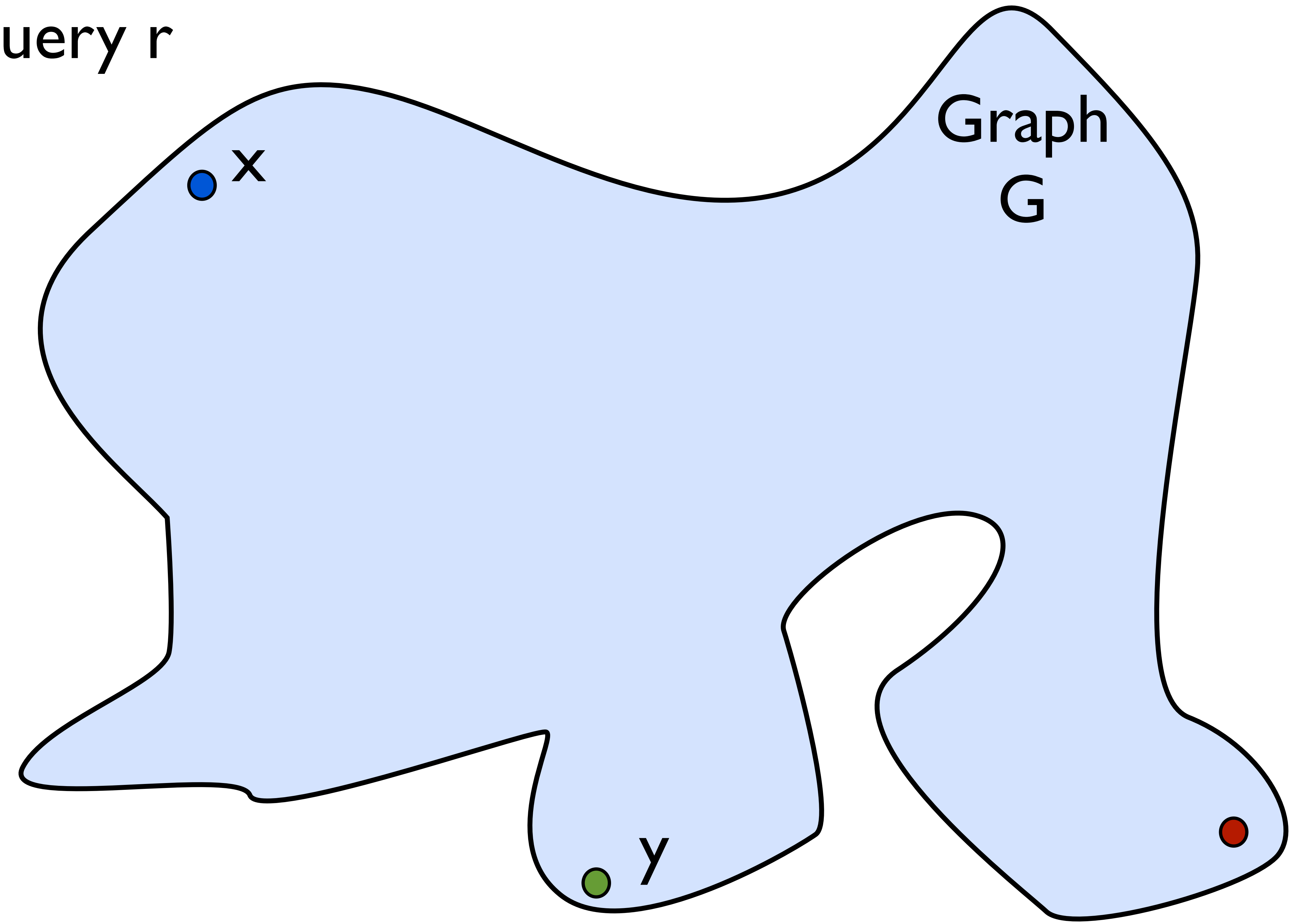
Reg Path Query r



More concrete

Reg Path Query r

Selects
(,)
instead of
(,)

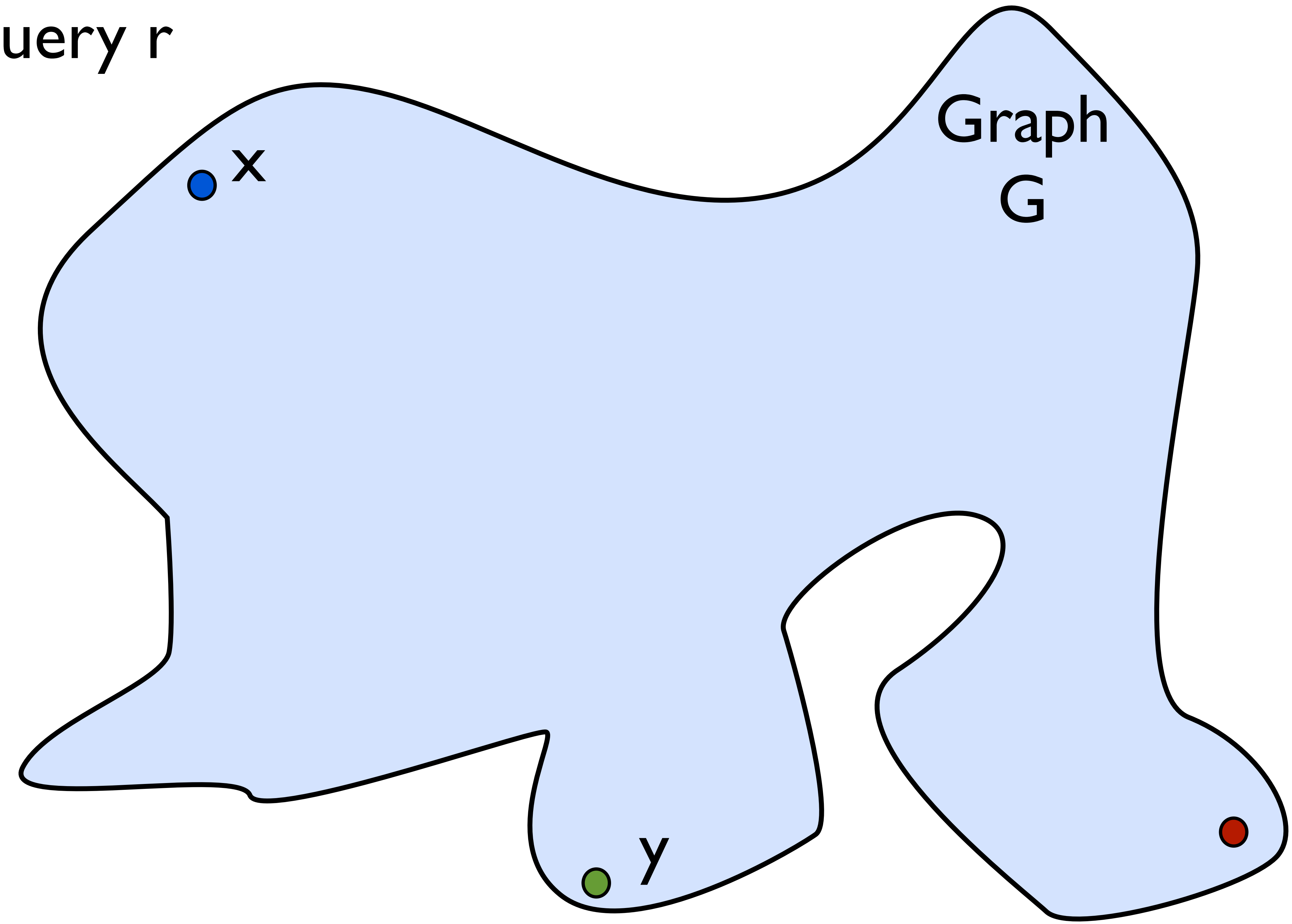


More concrete

Reg Path Query r

Selects
(● , ●)
instead of
(● , ●)

Why?



More concrete

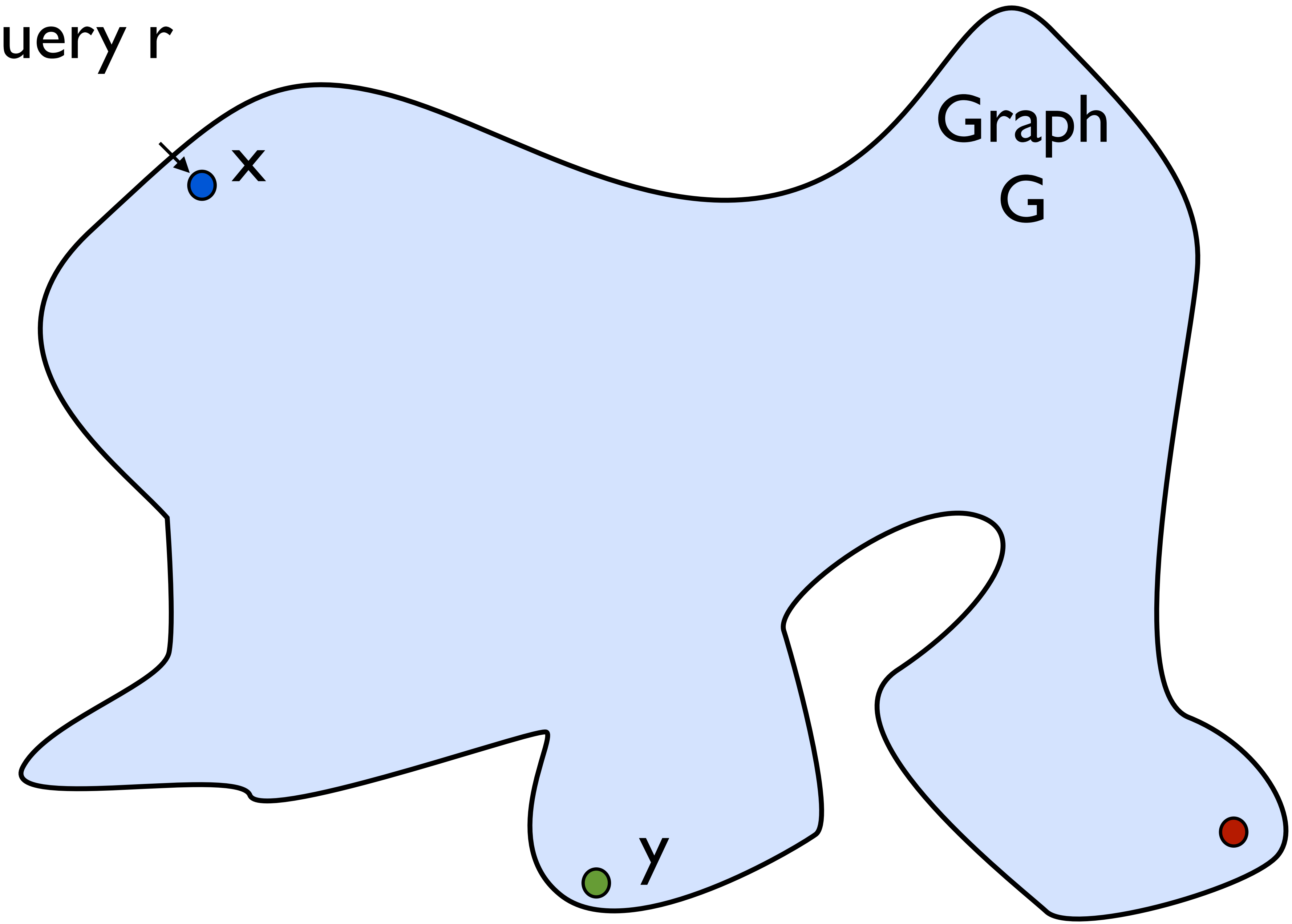
Reg Path Query r

Selects

(,)

instead of

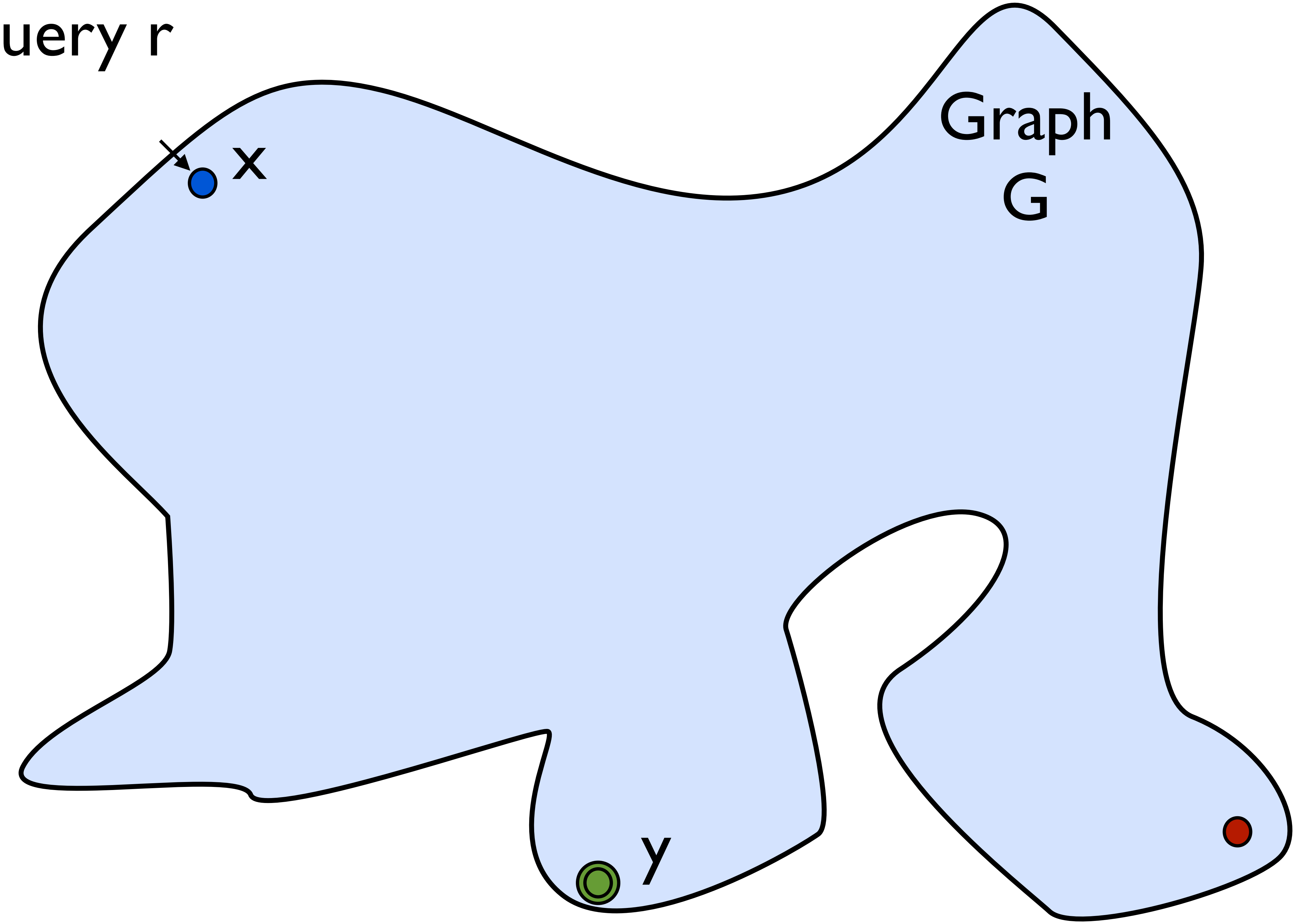
(,)



Why?

More concrete

Reg Path Query r



Selects
(,)
instead of
(,)

Why?

More concrete

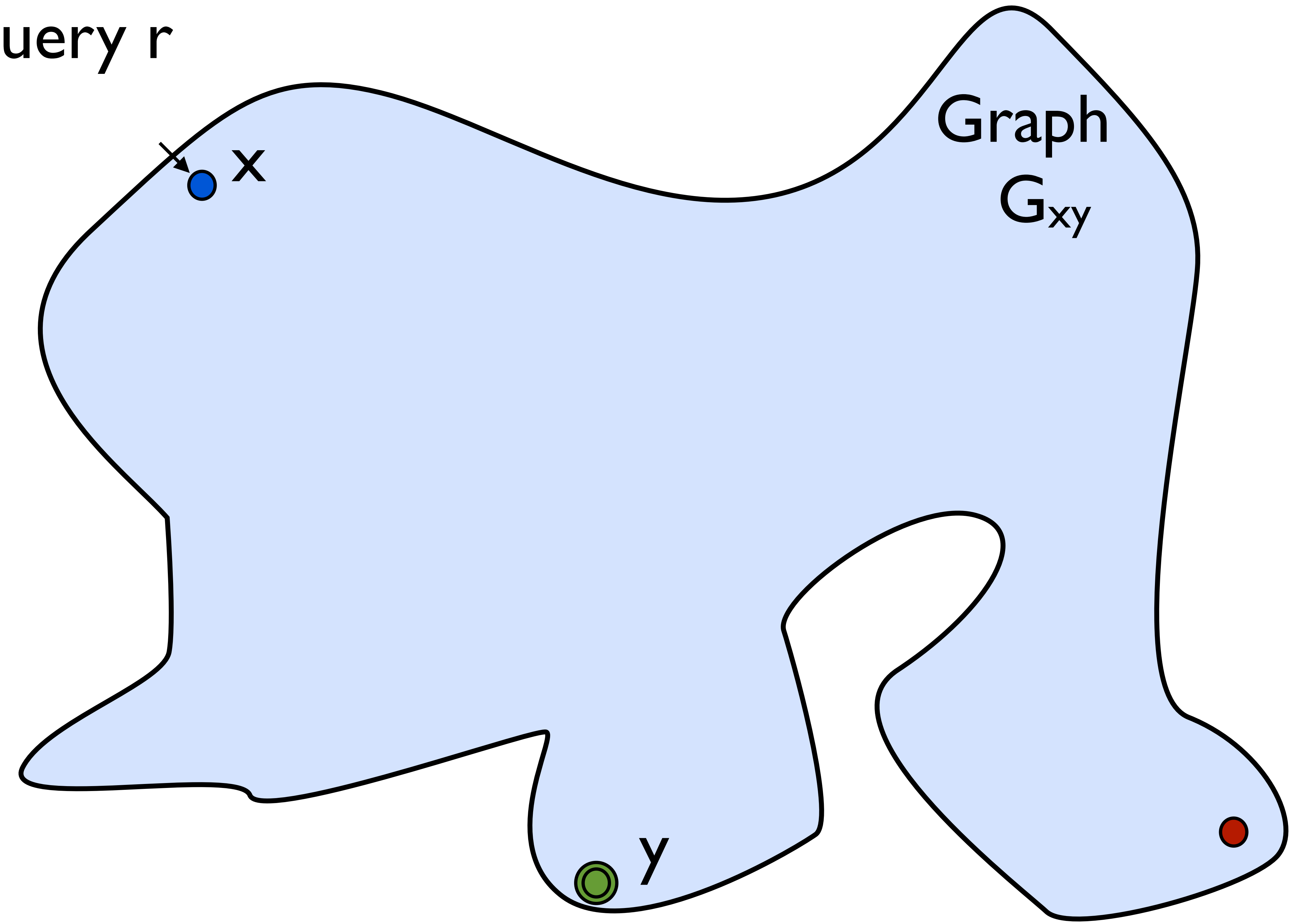
Reg Path Query r

Selects

(,)

instead of

(,)

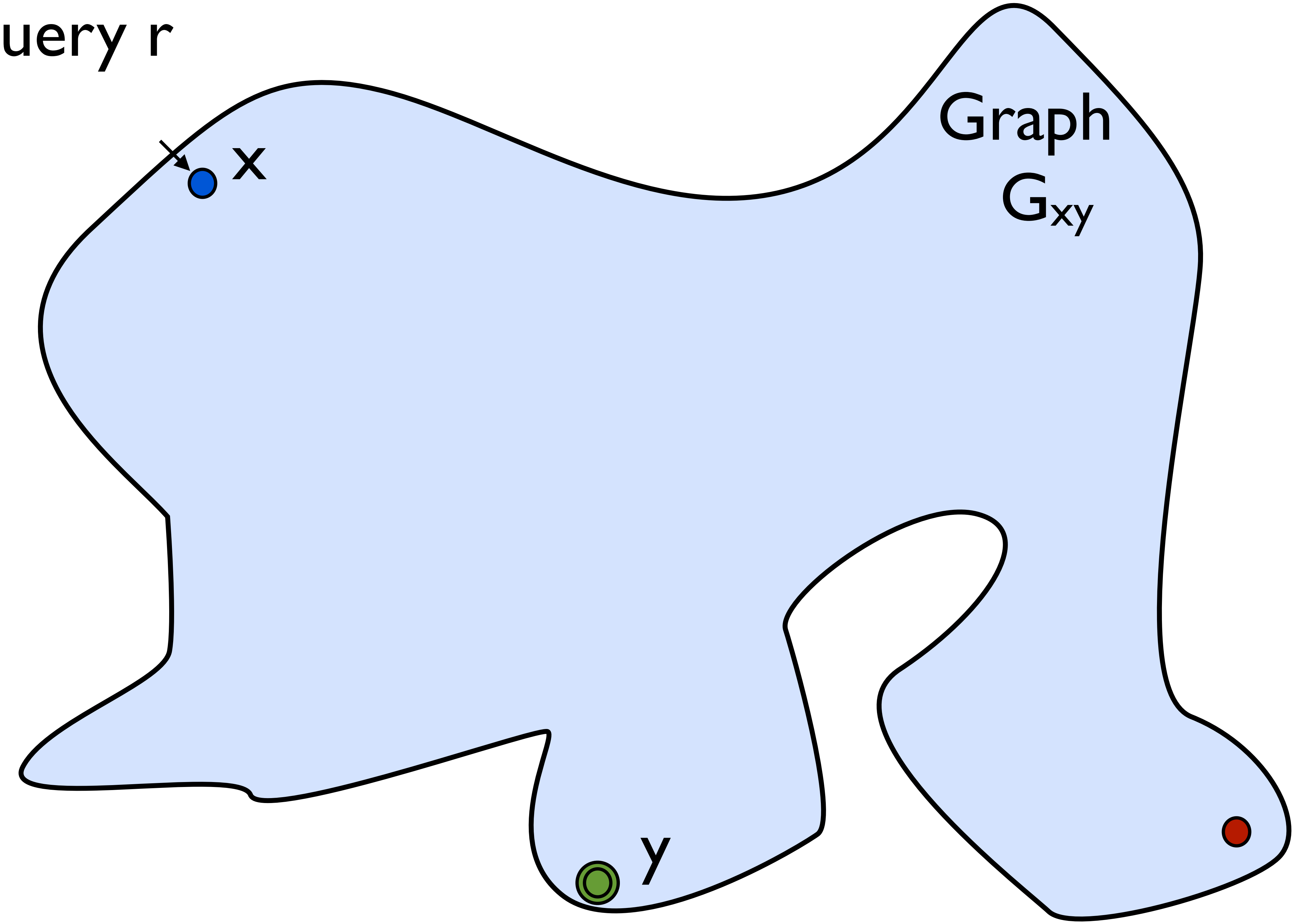


Why?

More concrete

Reg Path Query r

Selects
(,)
instead of
(,)



Why?

Because $L(r)$ and $L(G_{xy})$ have empty intersection

This problem boils down to

Given

- regular word language I
- regular word language E

why is I disjoint from E?

This problem boils down to

Given

- regular word language I
- regular word language E

why is I disjoint from E?

which language do we choose for saying why?

This problem boils down to

Given

- regular word language I
- regular word language E

why is I disjoint from E?

which language do we choose for saying why?

Of course:

This problem boils down to

Given

- regular word language I
- regular word language E

why is I disjoint from E?

which language do we choose for saying why?

Of course:

The Ultimate human-understandable language
that explains why things go wrong

This problem boils down to

Given

- regular word language I
- regular word language E

why is I disjoint from E?

which language do we choose for saying why?

Of course:

The Ultimate human-understandable language
that explains why things go wrong

This problem boils down to

Given

- regular word language I
- regular word language E

why is I disjoint from E?

which language do we choose for saying why?

Of course:

The Ultimate human-understandable language
that explains why things go wrong

This problem boils down to

Given

- regular word language I
- regular word language E

why is I disjoint from E?

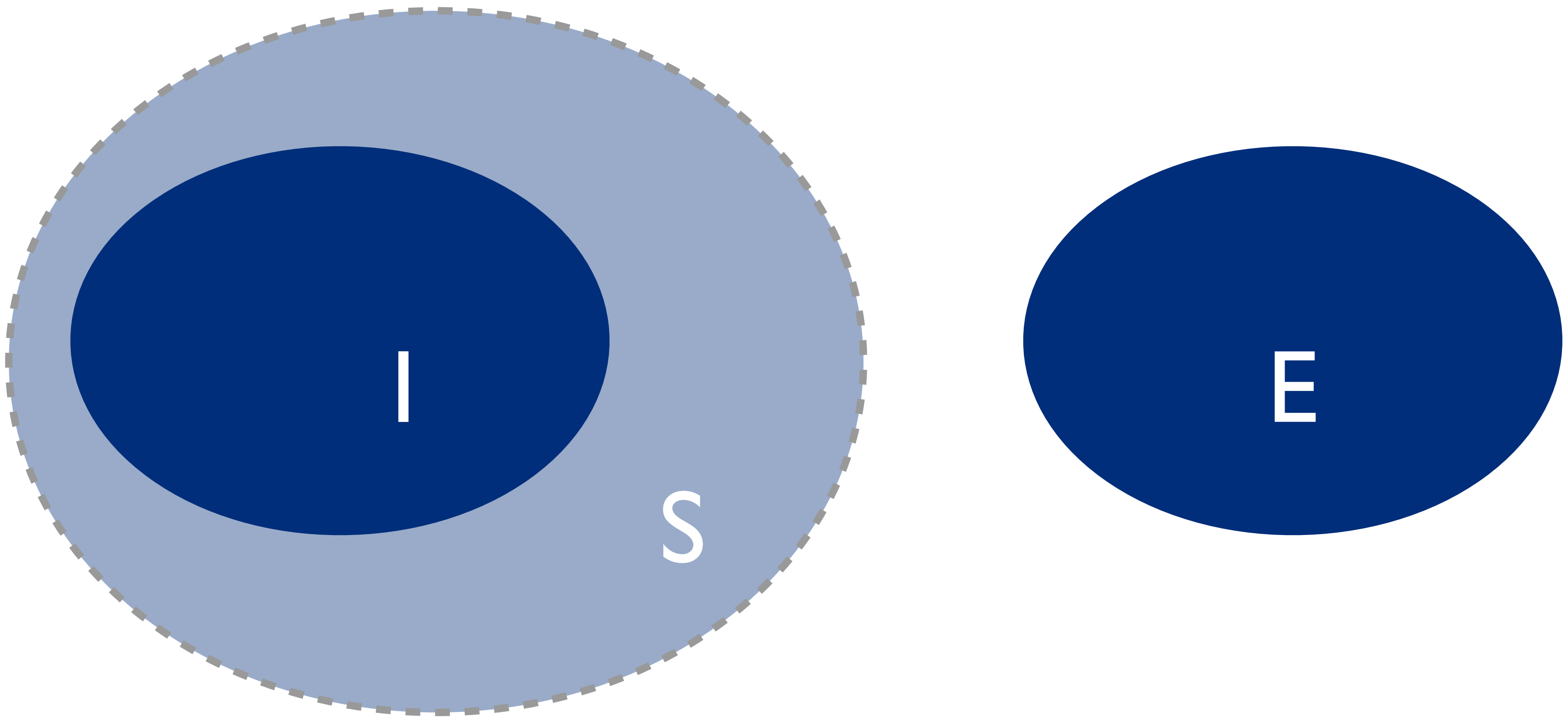
which language do we choose for saying why?

Of course:

The Ultimate human-understandable language
that explains why things go wrong

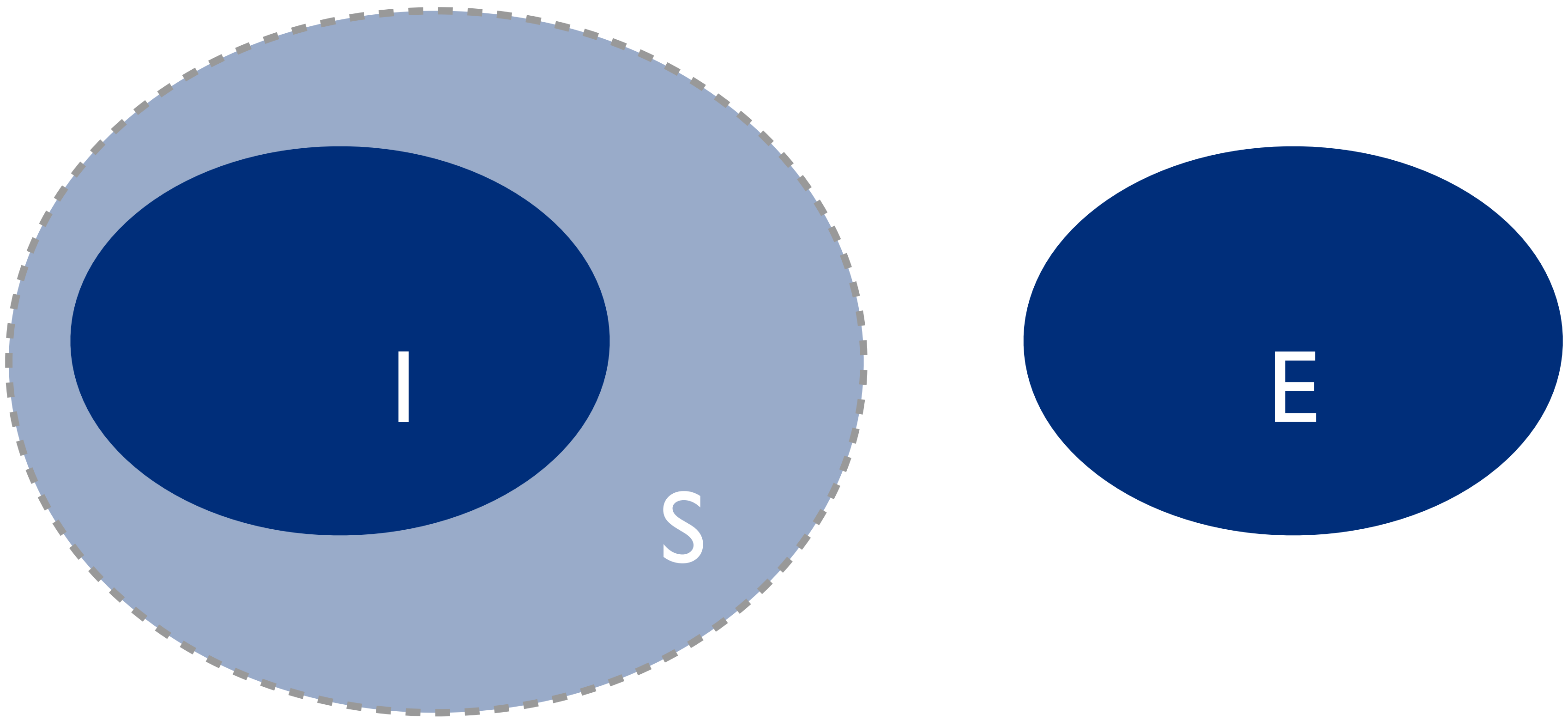
Separation

Separation



S *separates* I from E

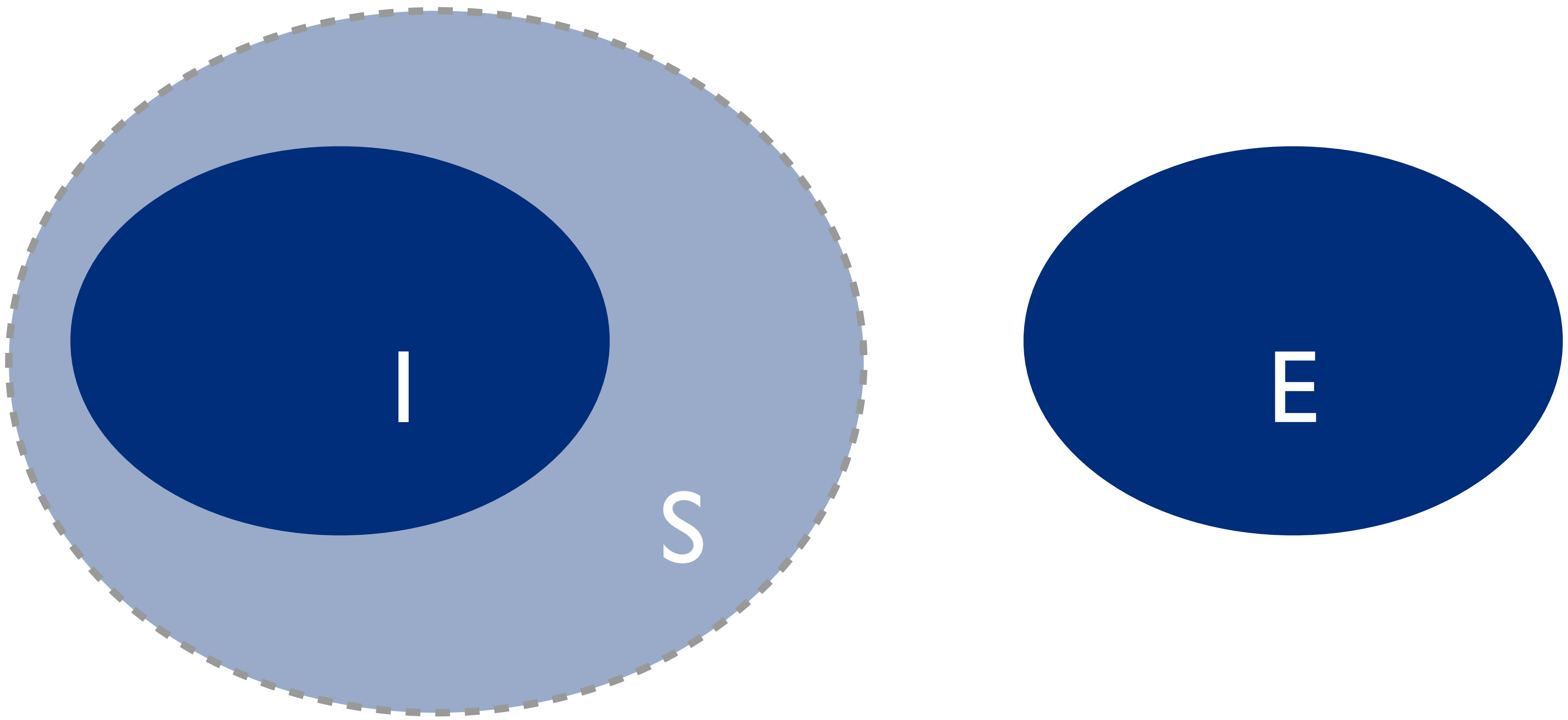
Separation



S *separates* I from E

I and E are *separable by* family F
if some S from F separates them

Separation

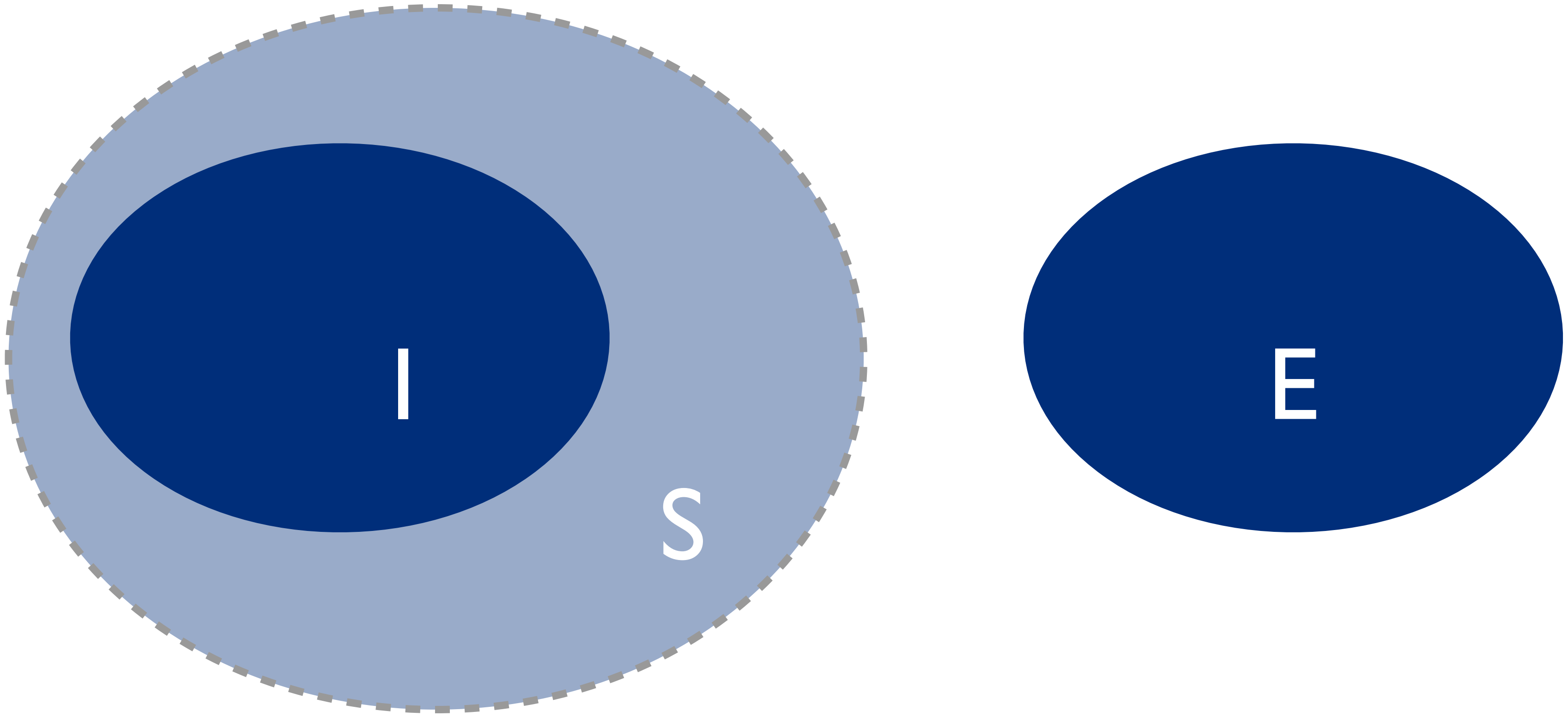


S *separates* I from E

Which $\mathcal{F}$?

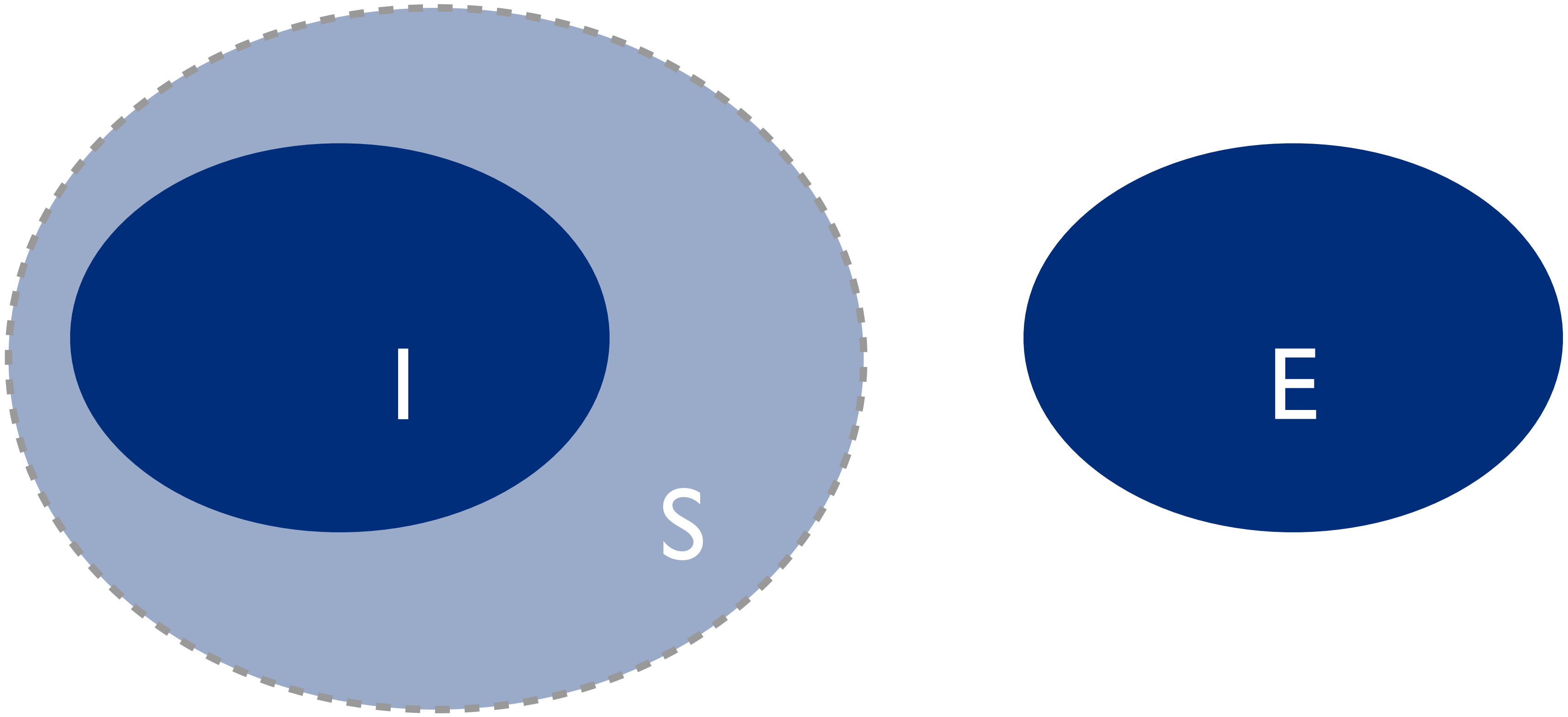
I and E are *separable by* family $\mathcal{F}$
if some S from $\mathcal{F}$ separates them

Separation



Here: S will come from families of

Separation



Here: S will come from families of

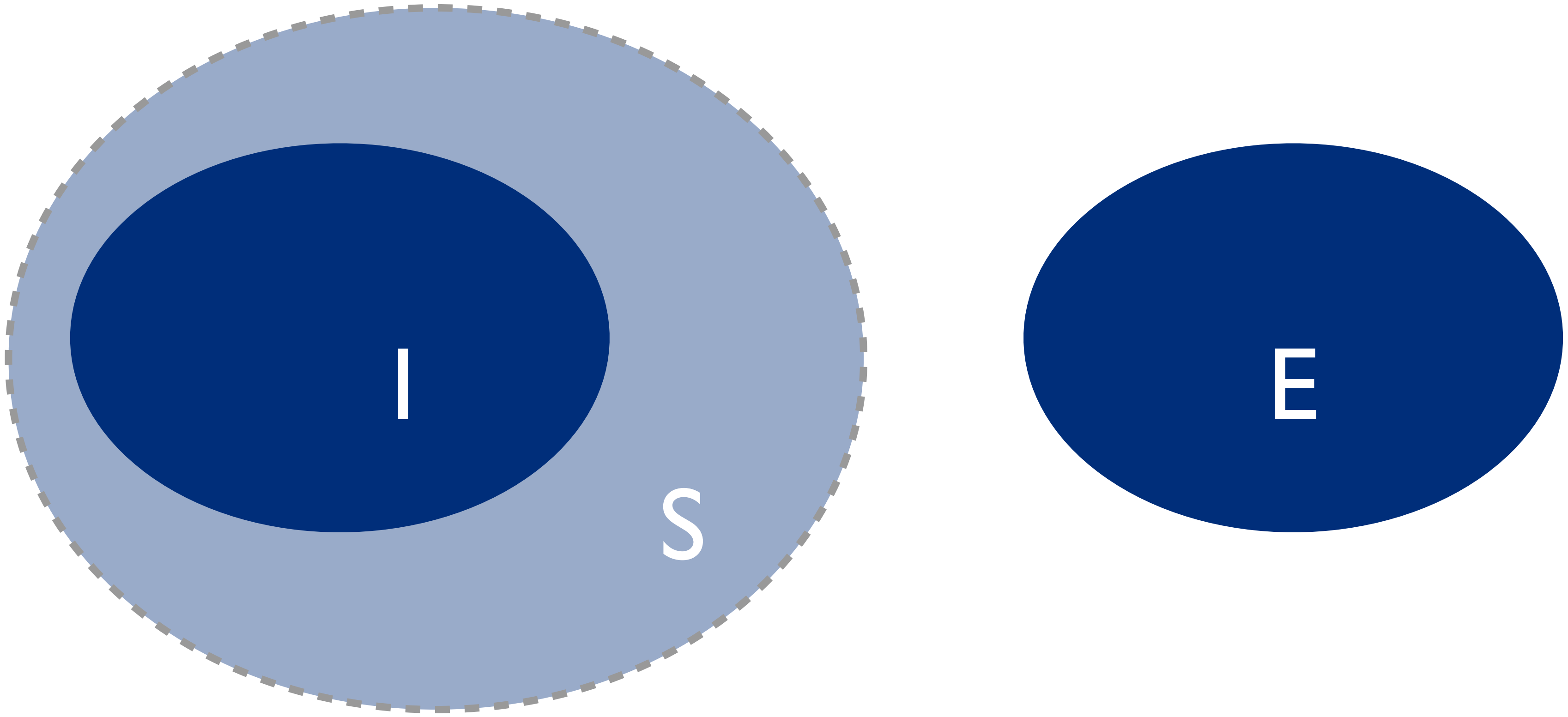
subword languages

...abc...

abc...

...abc

Separation



Here: S will come from families of

subword languages

...abc...

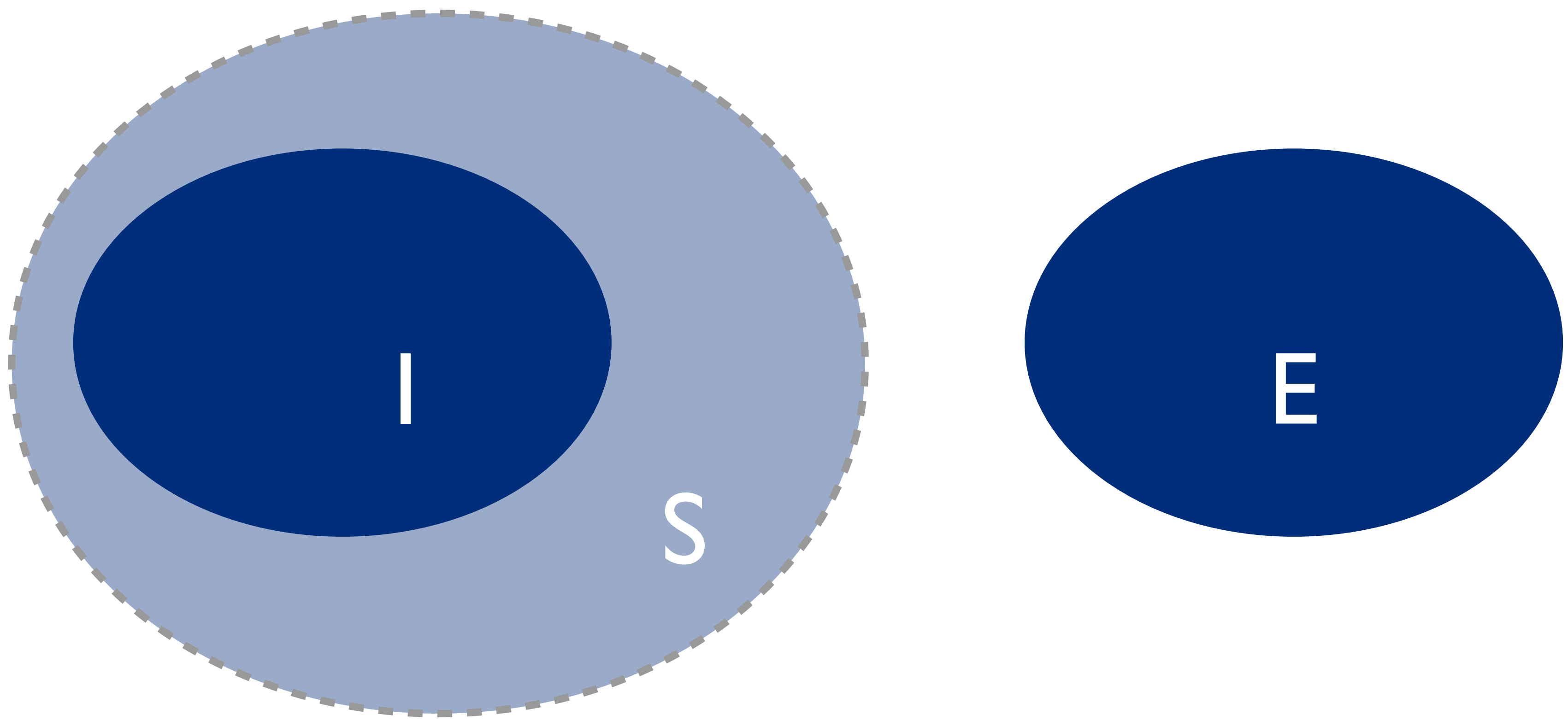
abc...

...abc

subsequence languages

...a...b...c...

Separation



Here: S will come from families of

subword languages

...abc...

abc...

...abc

subsequence languages

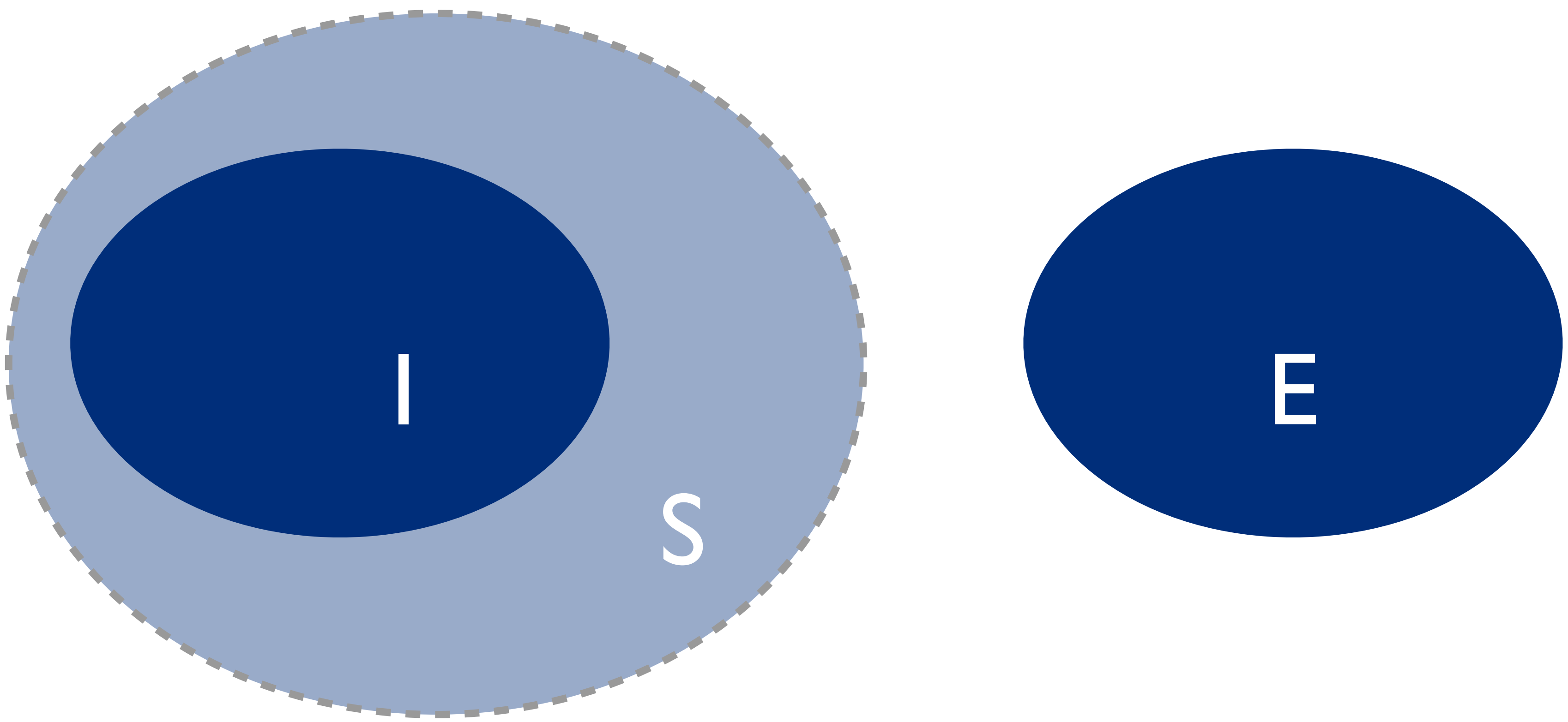
...a...b...c...

and combinations thereof

Main problem

Separability($\mathcal{F}$)

Given:	Regular languages I and E	(as NFAs)
Question:	Is I separable from E by some S in $\mathcal{F}$?	



So, here, we just decide separability
and our work is still very preliminary

Main problem

Separability($\mathcal{F}$)

Given:	Regular languages I and E	(as NFAs)
Question:	Is I separable from E by some S in $\mathcal{F}$?	

We will now look at different $\mathcal{F}$

Prefixes and Suffixes

A prefix language (over alphabet Σ)

is a language of the form

$$w\Sigma^*$$

for a word w

Prefixes and Suffixes

A prefix language (over alphabet Σ)

is a language of the form

$$w\Sigma^*$$

for a word w

It is a k-prefix language if $|w| \leq k$

Prefixes and Suffixes

A prefix language (over alphabet Σ)

is a language of the form

$$w\Sigma^*$$

for a word w

It is a k-prefix language if $|w| \leq k$

Theorem / Observation:

Separability($\mathcal{F}$) is in PTIME for the following $\mathcal{F}$:

- the prefix languages
- the k-prefix languages (for every k)

It remains in PTIME if we also allow

unions and boolean combinations

Prefixes and Suffixes

A prefix language (over alphabet Σ)

is a language of the form

$$w\Sigma^*$$

for a word w

It is a k-prefix language if $|w| \leq k$

Theorem / Observation:

Separability($\mathcal{F}$) is in PTIME for the following $\mathcal{F}$:

- the prefix languages
- the k-prefix languages (for every k)

It remains in PTIME if we also allow

unions and boolean combinations

Intuition: "local" explanations are easy to find

Subsequences

A subsequence language is a language of the form

$$\Sigma^*a_1\Sigma^*a_2\Sigma^*\ldots\Sigma^*a_n\Sigma^*$$

for letters $a_1,\ldots,a_n$

Subsequences

A subsequence language is a language of the form

$$\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \ldots \Sigma^* a_n \Sigma^*$$

for letters $a_1, \ldots, a_n$

Theorem [Czerwinski et al. ICALP13, van Rooijen et al. MFCS13]:

Separability(F) is in PTIME for the following F:

- boolean combinations of subsequence languages
- unions of subsequence languages

Subsequences

A subsequence language is a language of the form

$$\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \ldots \Sigma^* a_n \Sigma^*$$

for letters $a_1,\ldots,a_n$

Theorem [Czerwinski et al. ICALP13, van Rooijen et al. MFCS13]:

Separability(F) is in PTIME for the following F:

- boolean combinations of subsequence languages
- unions of subsequence languages

Intuition: Non-separability is some kind of reachability

Short Subsequences

A subsequence language is a language of the form

$$\Sigma^*a_1\Sigma^*a_2\Sigma^*\ldots\Sigma^*a_n\Sigma^*$$

for letters $a_1,\ldots,a_n$

It is a k-subsequence language if $n \leq k$

Short Subsequences

A subsequence language is a language of the form

$$\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \ldots \Sigma^* a_n \Sigma^*$$

for letters $a_1,\ldots,a_n$

It is a k-subsequence language if $n \leq k$

Theorem

Separability(F) is

- NP-complete for k-subsequence languages
- NP-hard / in Π_2^P for unions of k-subsequence languages
- coNP-complete for positive combinations
- coNP-hard / in NEXPTIME for bool combinations

(k is part of the input)

Short Subsequences

A subsequence language is a language of the form

$$\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \ldots \Sigma^* a_n \Sigma^*$$

for letters $a_1,\ldots,a_n$

It is a k-subsequence language if $n \leq k$

Theorem

Separability(F) is

- NP-complete for k-subsequence languages
- NP-hard / in Π_2^P for unions of k-subsequence languages
- coNP-complete for positive combinations
- coNP-hard / in NEXPTIME for bool combinations

(k is part of the input)

Short Subsequences

Reduction from SAT

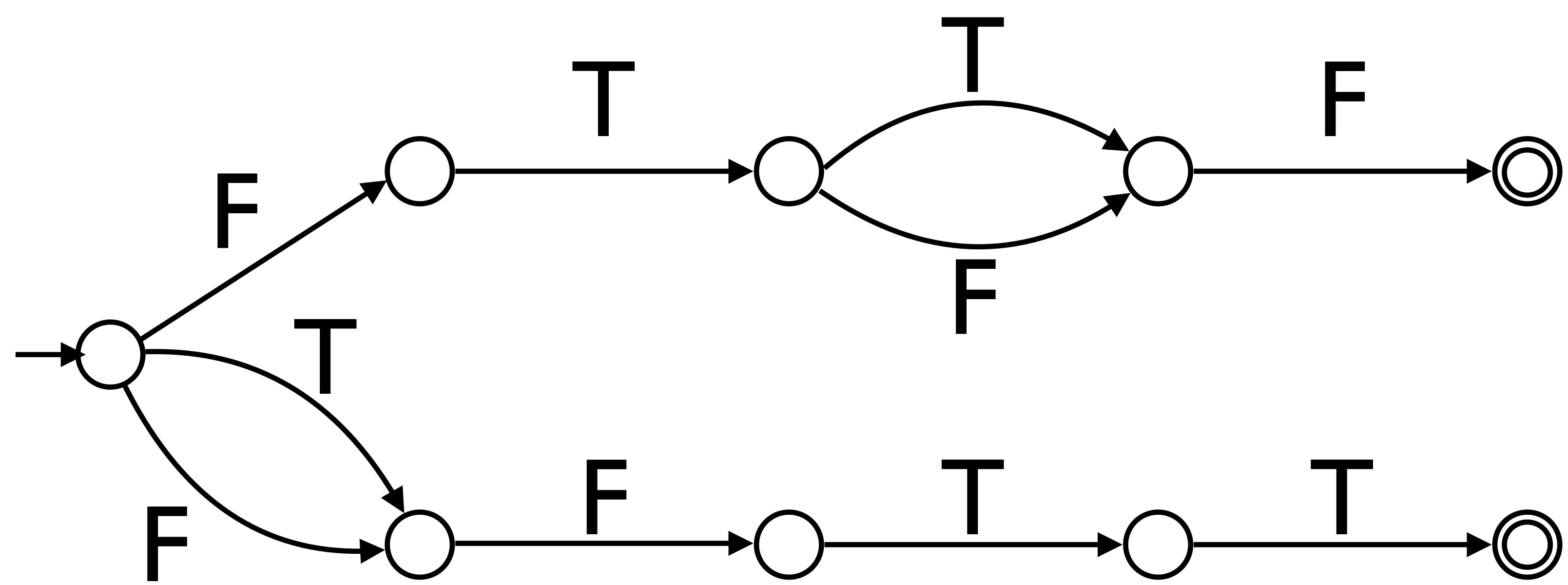
Let $\varphi = (x_1 \lor {\sim}x_2 \lor x_4)$ and $(x_2 \lor {\sim}x_3 \lor {\sim}x_4)$

Short Subsequences

Reduction from SAT

Let $\varphi = (x_1 \lor \sim x_2 \lor x_4)$ and $(x_2 \lor \sim x_3 \lor \sim x_4)$

Let E =

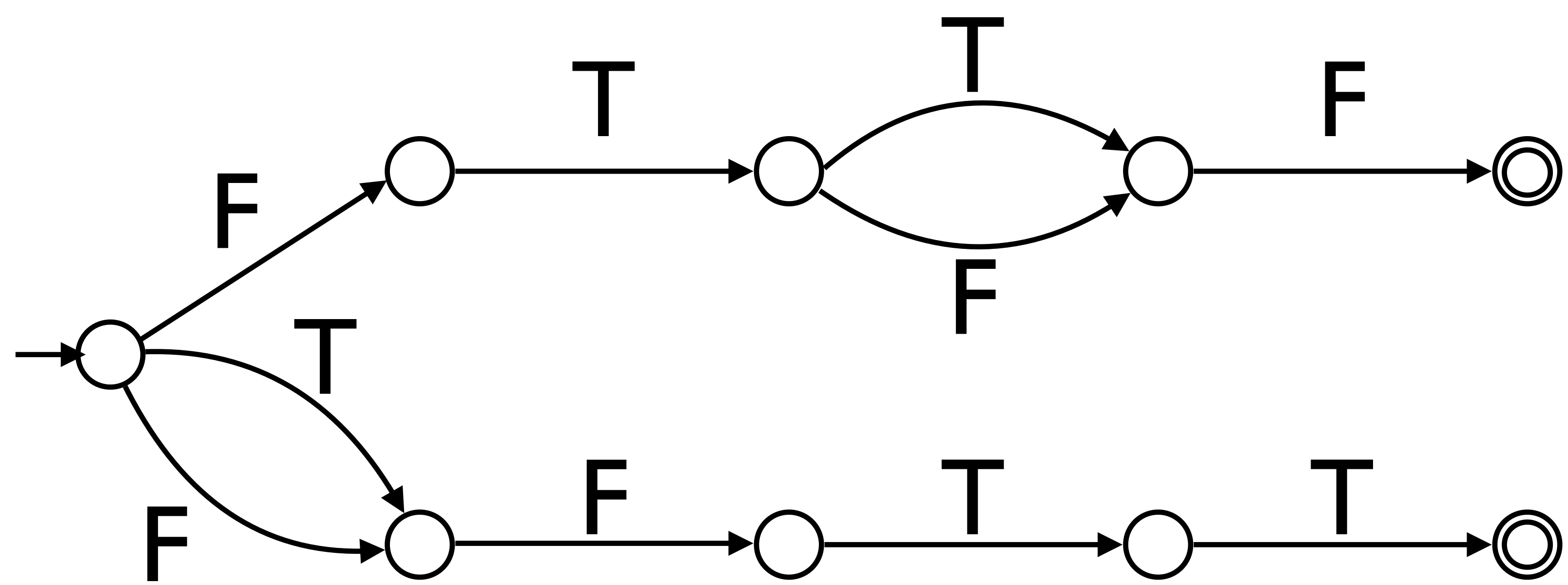


Short Subsequences

Reduction from SAT

Let $\varphi = (x_1 \vee \sim x_2 \vee x_4)$ and $(x_2 \vee \sim x_3 \vee \sim x_4)$

Let E =

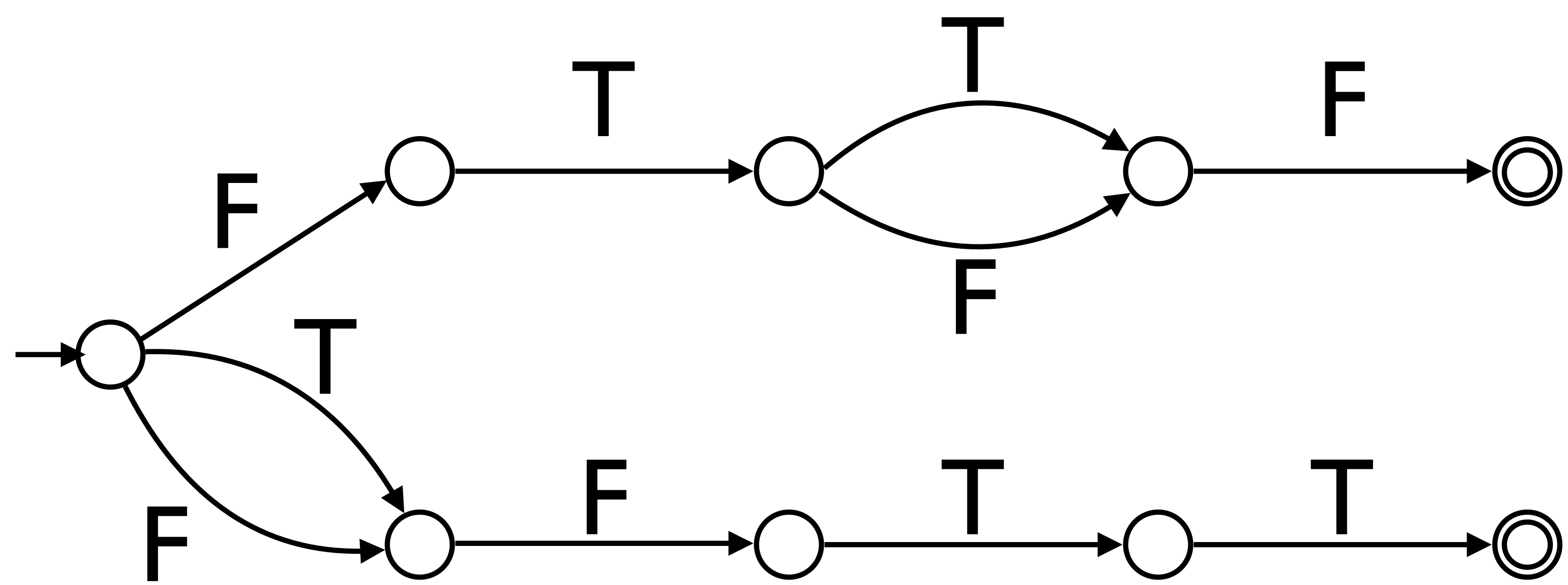


Let I = TFTFTFTF

Short Subsequences

Reduction from SAT

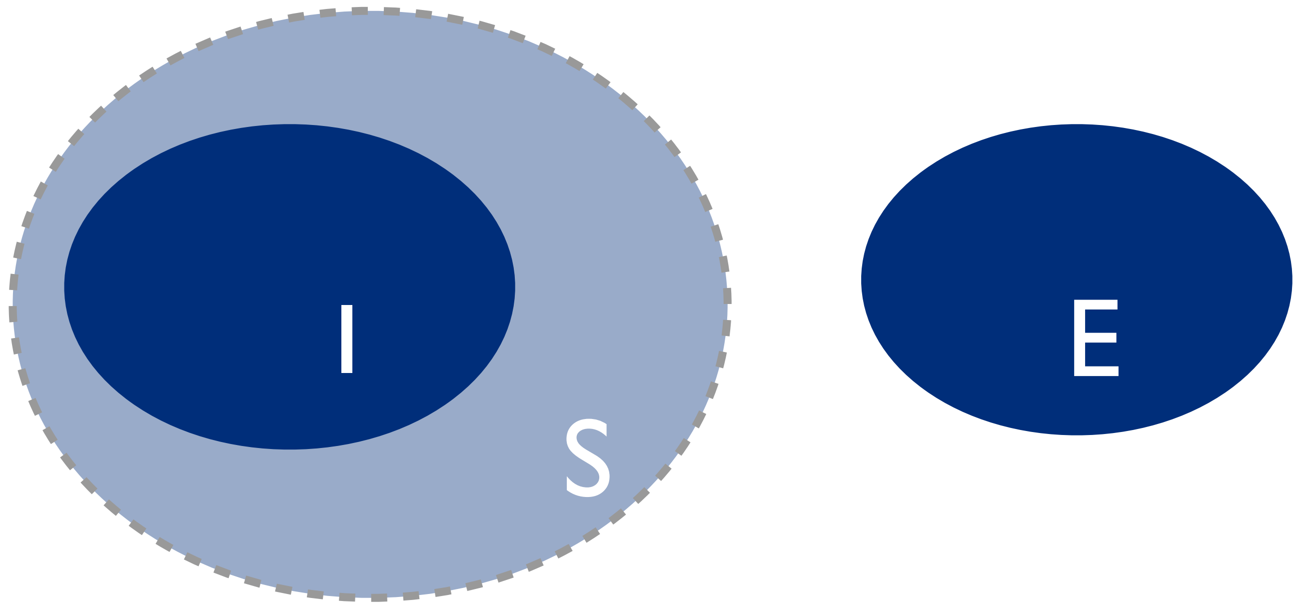
Let $\varphi = (x_1 \vee \sim x_2 \vee x_4)$ and $(x_2 \vee \sim x_3 \vee \sim x_4)$

Let E = 

Let I = TFTFTFTF

I is separable from E
by 4-subsequence language
iff
$L(E) \neq (T+F)(T+F)(T+F)(T+F)$

Subsequences: Restricting I and E

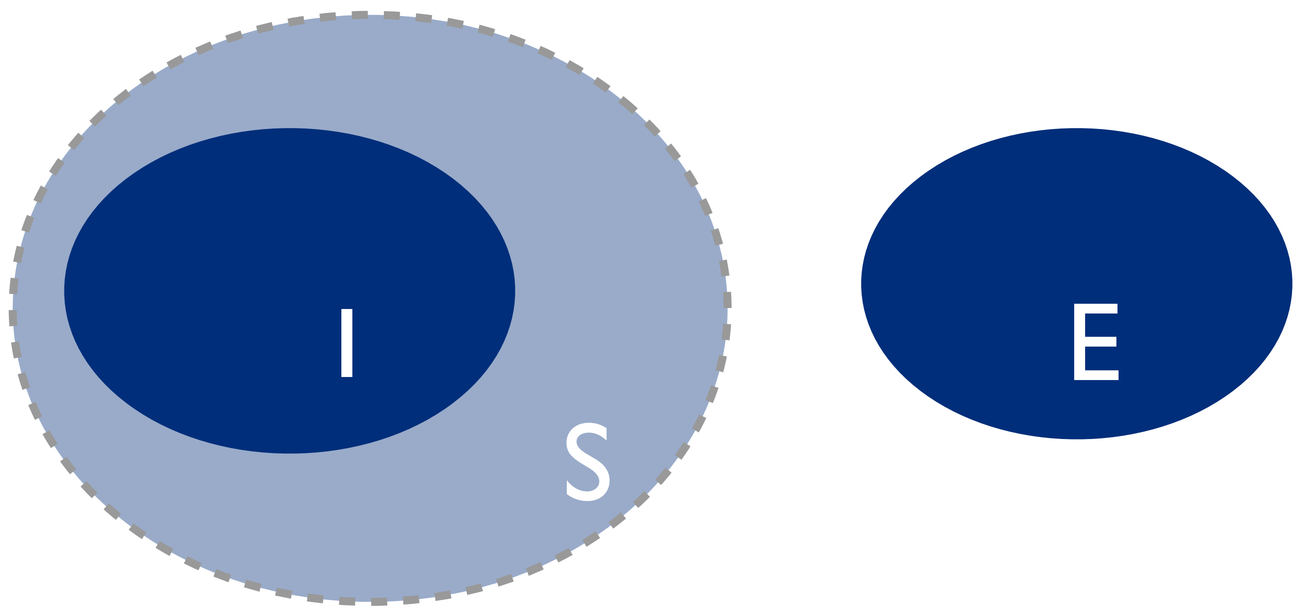


What happens if we restrict I or E?

If E has a constant-size core-approximation, then separability of I from E is in PTIME for

- k-subsequence languages and
- unions / intersections / positive combinations of k-subsequence languages

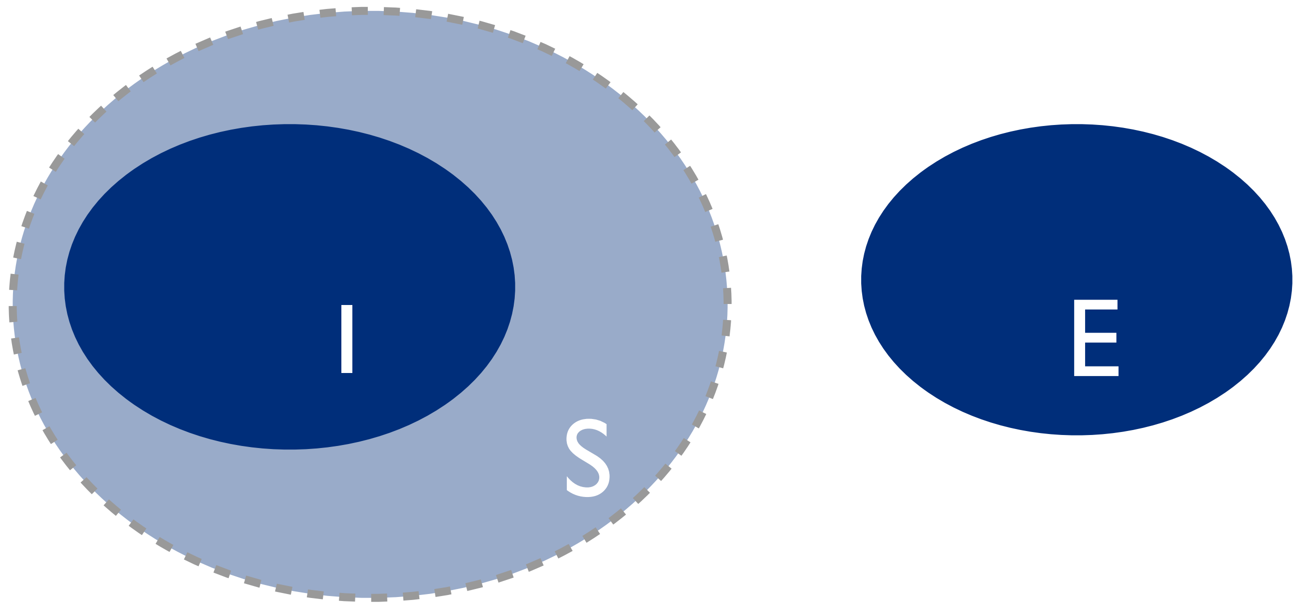
Subsequences: Restricting I and E



Core-approximation of an NFA:

- Collapse all strongly connected components
- Perform bisimulation minimization

Subsequences: Restricting I and E



If E has a constant-size core-approximation, then separability of I from E is in PTIME for

- k-subsequence languages

This technique can be extended to show tractable separability by k-subsequences of constant-length words

$\ldots a_1 b_1 \ldots a_2 b_2 \ldots \quad . \; . \; . \quad \ldots a_k b_k \ldots$

Recap and Other Results

The complexity of separability by

Recap and Other Results

The complexity of separability by

- Prefixes and suffixes: tractable

Recap and Other Results

The complexity of separability by

- Prefixes and suffixes: tractable
- Subsequences:

Recap and Other Results

The complexity of separability by

- Prefixes and suffixes: tractable
- Subsequences:
  - tractable if the length of subsequence doesn't matter

Recap and Other Results

The complexity of separability by

- Prefixes and suffixes: tractable
- Subsequences:
  - tractable if the length of subsequence doesn't matter
  - NP- / coNP-hard if the max length k is in the input

Recap and Other Results

The complexity of separability by

- Prefixes and suffixes: tractable
- Subsequences:
  - tractable if the length of subsequence doesn't matter
  - NP- / coNP-hard if the max length k is in the input
- Subwords:

Recap and Other Results

The complexity of separability by

- Prefixes and suffixes: tractable
- Subsequences:
  - tractable if the length of subsequence doesn't matter
  - NP- / coNP-hard if the max length k is in the input
- Subwords:
  - separability by a subword language: tractable

Recap and Other Results

The complexity of separability by

- Prefixes and suffixes: tractable
- Subsequences:
  - tractable if the length of subsequence doesn't matter
  - NP- / coNP-hard if the max length k is in the input
- Subwords:
  - separability by a subword language: tractable
  - unions, intersections, positive-, boolean combinations of k-subword languages: from coNP to PSPACE-hard

Recap and Other Results

The complexity of separability by

- Prefixes and suffixes: tractable
- Subsequences:
  - tractable if the length of subsequence doesn't matter
  - NP- / coNP-hard if the max length k is in the input
- Subwords:
  - separability by a subword language: tractable
  - unions, intersections, positive-, boolean combinations of k-subword languages: from coNP to PSPACE-hard

A promising case seems to be
k-subsequences of constant-length subwords

Concluding Remarks

Separation is a very general and exciting problem:

Why is language 1 disjoint from language 2?

Concluding Remarks

Separation is a very general and exciting problem:

Why is language 1 disjoint from language 2?

It's been a research topic in language theory for a while now but seems to be gaining momentum nowadays

Concluding Remarks

We just scratched the surface

Concluding Remarks

We just scratched the surface

There is a huge body of interesting remaining questions:

Concluding Remarks

We just scratched the surface

There is a huge body of interesting remaining questions:

- Which separators can we efficiently compute?

Concluding Remarks

We just scratched the surface

There is a huge body of interesting remaining questions:

- Which separators can we efficiently compute?
- Which other classes of separators to consider?

Concluding Remarks

We just scratched the surface

There is a huge body of interesting remaining questions:

- Which separators can we efficiently compute?
- Which other classes of separators to consider?
- What are good measures for "simplicity" of a separator?

Concluding Remarks

We just scratched the surface

There is a huge body of interesting remaining questions:

- Which separators can we efficiently compute?
- Which other classes of separators to consider?
- What are good measures for "simplicity" of a separator?
- What will work in practice?

Concluding Remarks

We just scratched the surface

There is a huge body of interesting remaining questions:

- Which separators can we efficiently compute?
- Which other classes of separators to consider?
- What are good measures for "simplicity" of a separator?
- What will work in practice?

Interesting related question: Why is a result in the answer?

Thank you!

Questions?