ACM SIGMOD/PODS 2016

June 26, San Francisco, CA, USA

Ronald Fagin Special Event

Database Principles in Text Analytics

Benny Kimelfeld

Technion Data & Knowledge Lab

Faculty of Computer Science, Technion, Israel



CV


Technion
Israel Institute of Technology
2015
LogicBlox
2014
IBM
Ron
(Ph.D.)
Laura Chiticariu
Phokion Kolaitis
Sekar Krishnamurthy
Fred Reiss
Sriram Raghavan
Yunyao Li
Shiv Vaithyanathan
2009
האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM
2000

What did I do at Almaden?







For one, permanent committee member of the annual **CS picnic**







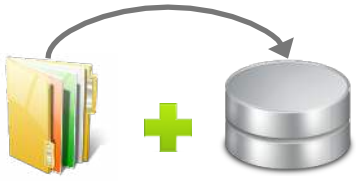
Thanks Almaden for providing pictures!

Outline

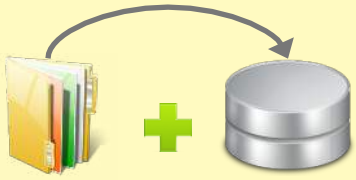


- Enterprise Search
- Information Extraction
- Prioritized Repairing

Concepts of Search over Structured Data

Query	Data	Answer	Typical
Keywords (sequence of terms, no restrictions)	(Structured) DB 	Connected set of tuples/items	*Data = graph; answer = subtree; kws = leaves*
	DB + schema 	DB query	*Answer = CQ that connects the keywords*
	Tree data (XML) 	Tree Node	*Each subtree treated as a separate document*
	Docs + aux. DB 	Document	*DB indexes entities and relationships inside the documents*

Concepts of Search over Structured Data

Query	Data	Answer	
Keywords (sequence of terms, no restrictions)	(Structured) DB 	Connected set of tuples/items	Explored in my PhD w/ Shuky Sagiv
	DB + schema 	DB query	
	Tree data (XML) 	Tree Node 	
	Docs + aux. DB 	Document 	Work w/ Ron @Almaden

Enterprise Search Projects @Almaden

- OmniFind
  - Personal email search





- Gumshoe
  - Enterprise (internal Web) search

Example: Email Search

from sara john number | Search

Interpretation:

Find emails that contain the words "from" "sara" "john" and "number"



Interpretation:

Find emails from Sara, where some phone# and "john" is included



Interpretation:

Find emails from Sara, s.t. the phone# of the person "John" is included



Re: Profile Error Again

From: Sara Shackleton <sara@enron.com>

Sent: 05/16/2001 at 15:32

Emma, **person** **phone**

Please call John at 713-853-4145.

person-phone

Sara Shackleton **person**

Enron North America Corp.

1400 Smith Street, EB 3801a

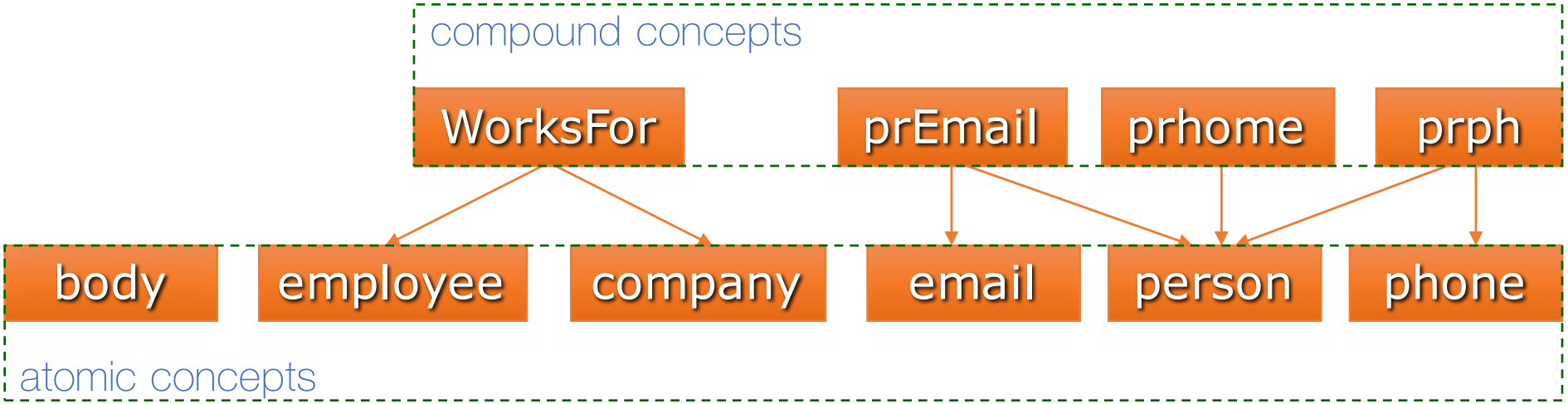
Houston, Texas 77002

713-853-5620 **phone**

sara@enron.com **email**

signature

Search Database Schema



A schema is a partially ordered set of concepts

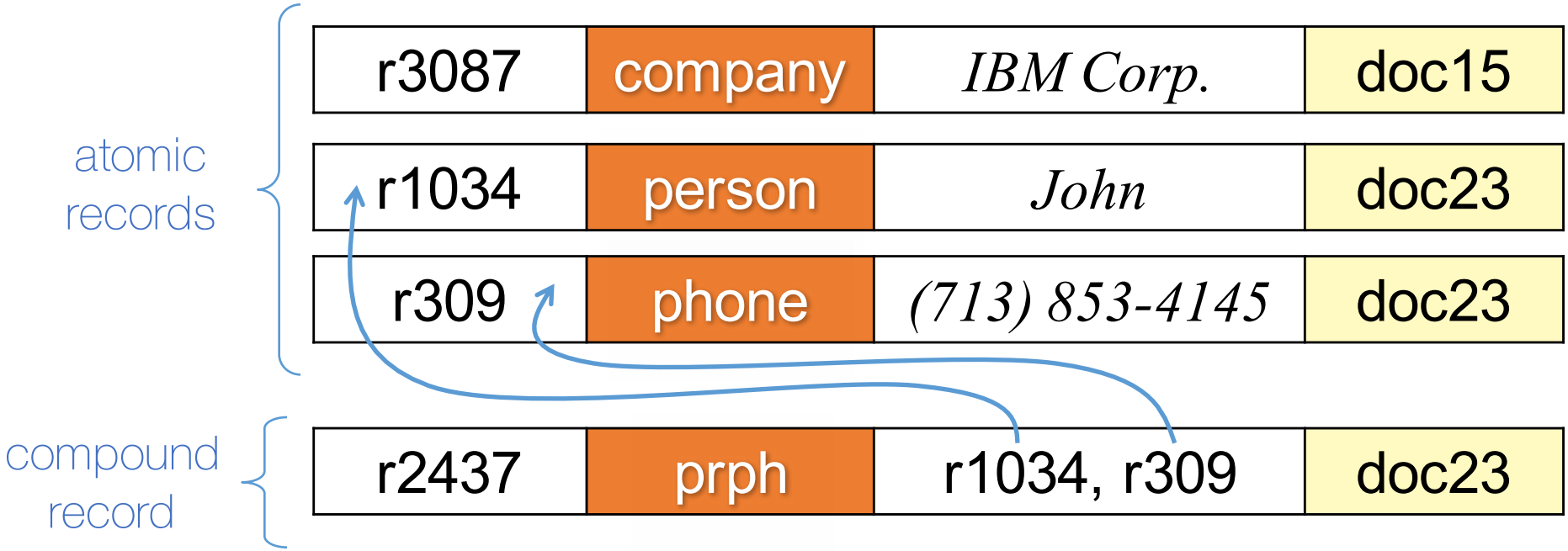
+ subtyping

employee ⊆ person

Database: Instances of Concepts

A database is a set of records

(atomic & compound records)



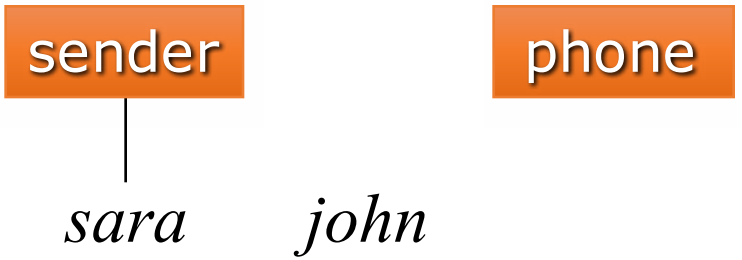
From Search Queries to DB Queries

from sara john number | Search

from sara john number

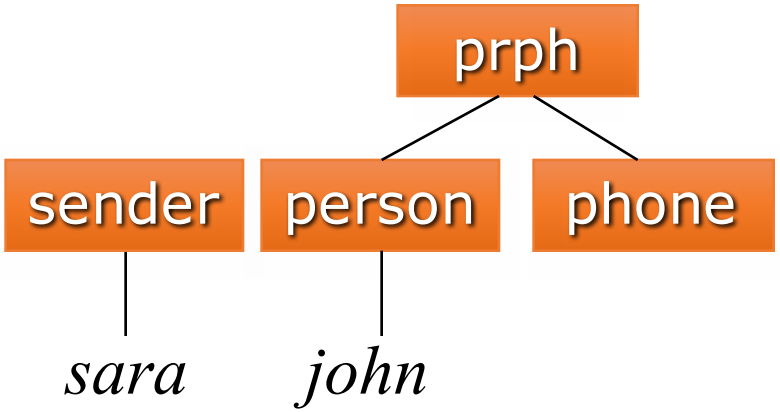


Each of the four keywords should occur in the document





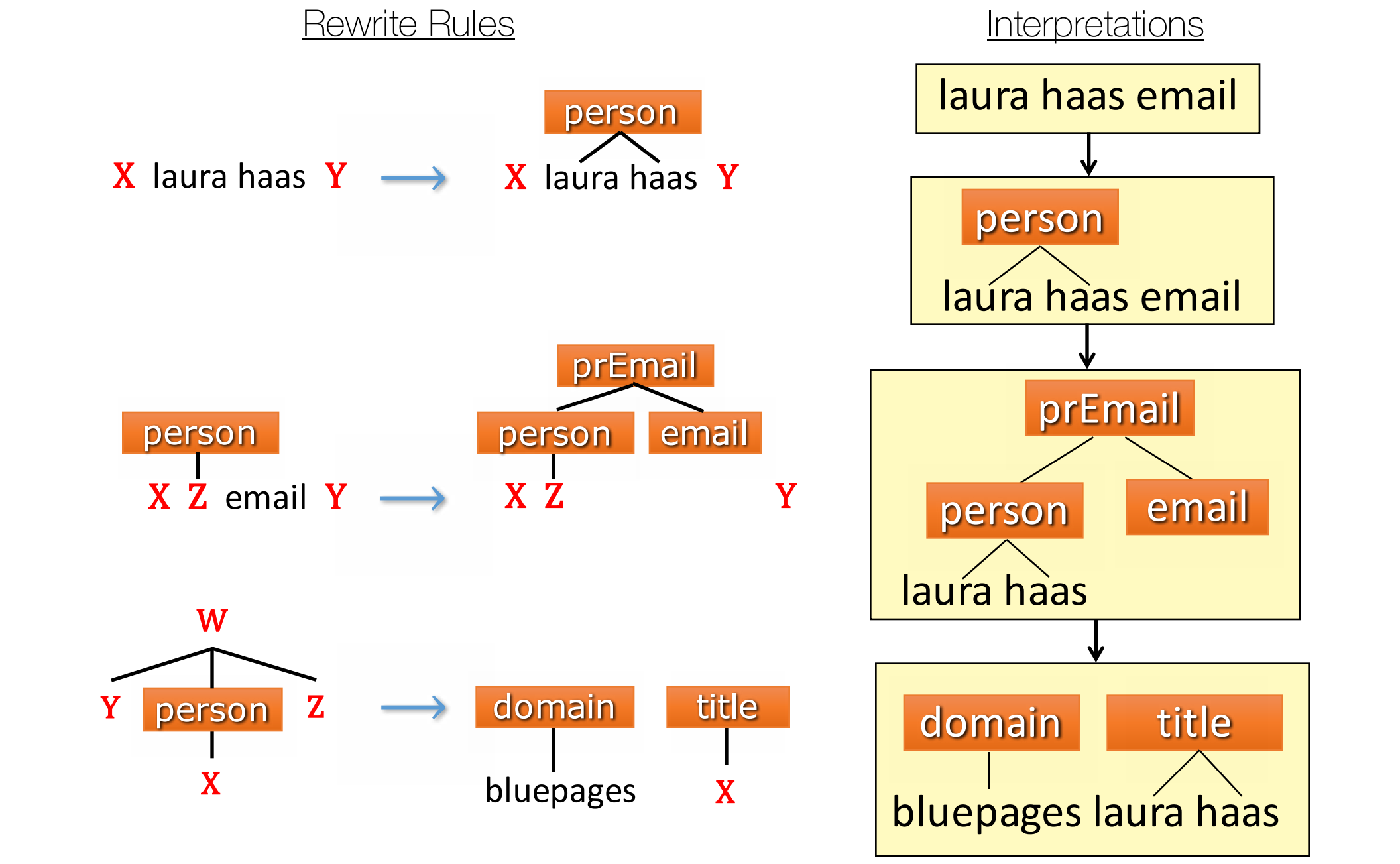
The sender is “sara, “john” is included, some phone# is included





The sender is “sara,” person “john” is included along w/ his phone#

Rewrite Rules



Research

- **Framework** [Fagin, K, Li, Raghavan, Vaithyanathan, PODS10]
  - "Search database systems"
  - Specificity (or containment) of interpretations
  - How to produce (top-specific, nonempty) interpretations?
- **Convergence** [Fagin, K, Li, Raghavan, Vaithyanathan, PODS11]
  - *How to apply rewrite rules to the search query?*
  - Simple way: each rule applied once, predefined order
  - Thorough way: least fixpoint (apply repeatedly)
    - Problem: "bad" rule sets lead to non-termination
      - Real problem: termination is undecidable
    - Robust & tractable safety guarantees termination

Sources of Auxiliary Data

SystemT

- Information extraction *Next topic*






  - Signature, person, phone, person-phone, …
- Domain knowledge
  - Email search: email headers (metadata), user's address book, etc.
  - Enterprise search: business data, HR data, etc.
  - Online store search: product database, etc.
- Global knowledge
  - WordNet, DBPedia, YAGO, GeoNames, …

Outline

- Enterprise Search
- Information Extraction
- Prioritized Repairing

Information Extraction (IE)

data-in-text → data-in-db
(unstructured) (structured)

"Information Extraction (IE) is the name given to any process which selectively ***structures and combines data*** *which is found, explicitly stated or implied,* ***in one or more texts****. The final output of the extraction process varies; in every case, however, it can be transformed so as to* ***populate some type of database****."*

J. Cowie and Y. Wilks., *Handbook of Natural Language Processing*, 2000

IE with IBM's SystemT

```
create view Caps as
extract regex /[A-Z](\w|-)+/ on D.text as name from Document D;

create view Last as
extract dictionary LastGaz on D.text as name from Document D;

create view CapsLast as
select  CombineSpans(C.name, L.name) as name
from    Caps C, Last L
where   FollowsTok(C.name, L.name, 0, 0);
...

create view PersonAll as
    (select R.name from FirstLast R) union all ...
     ... union all (select R.name from CapsLast R);

create view Person as select * from PersonAll R
consolidate on R.name using 'ContainedWithin';

output view Person;
```

"Regex formulas"

regex + join w/ previous views

union

projection

Cleaning

[Chiticariu, Krishnamurthy, Li, Raghavan, Reiss, Vaithyanathan, ACL 2010]

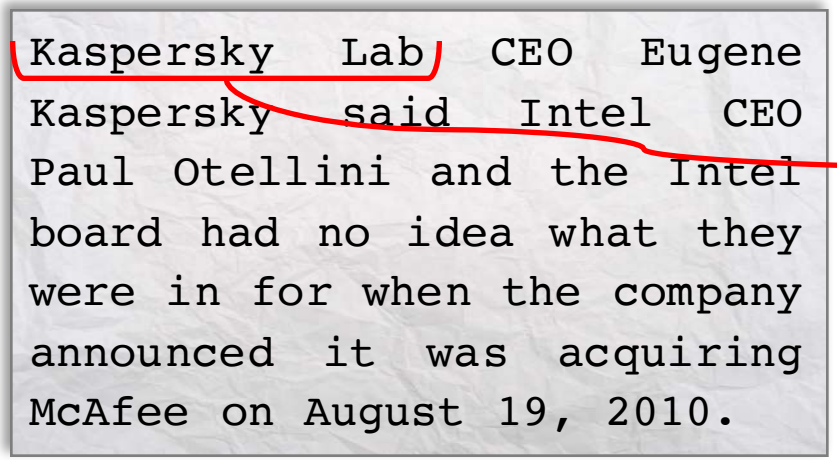
Document Spanners

[Fagin, K, Reiss, Vansummeren, JACM15]

Document Spanner: a function that maps every doc (string) into a relation over the doc's spans

More formally:

- Finite alphabet S of symbols
- A spanner maps each doc. $d \in S^*$ into a relation over the spans $[i,j)$ of d
- The relation has a fixed signature (set of attributes)
  - The attributes come from an infinite domain of variables x, y, z, …

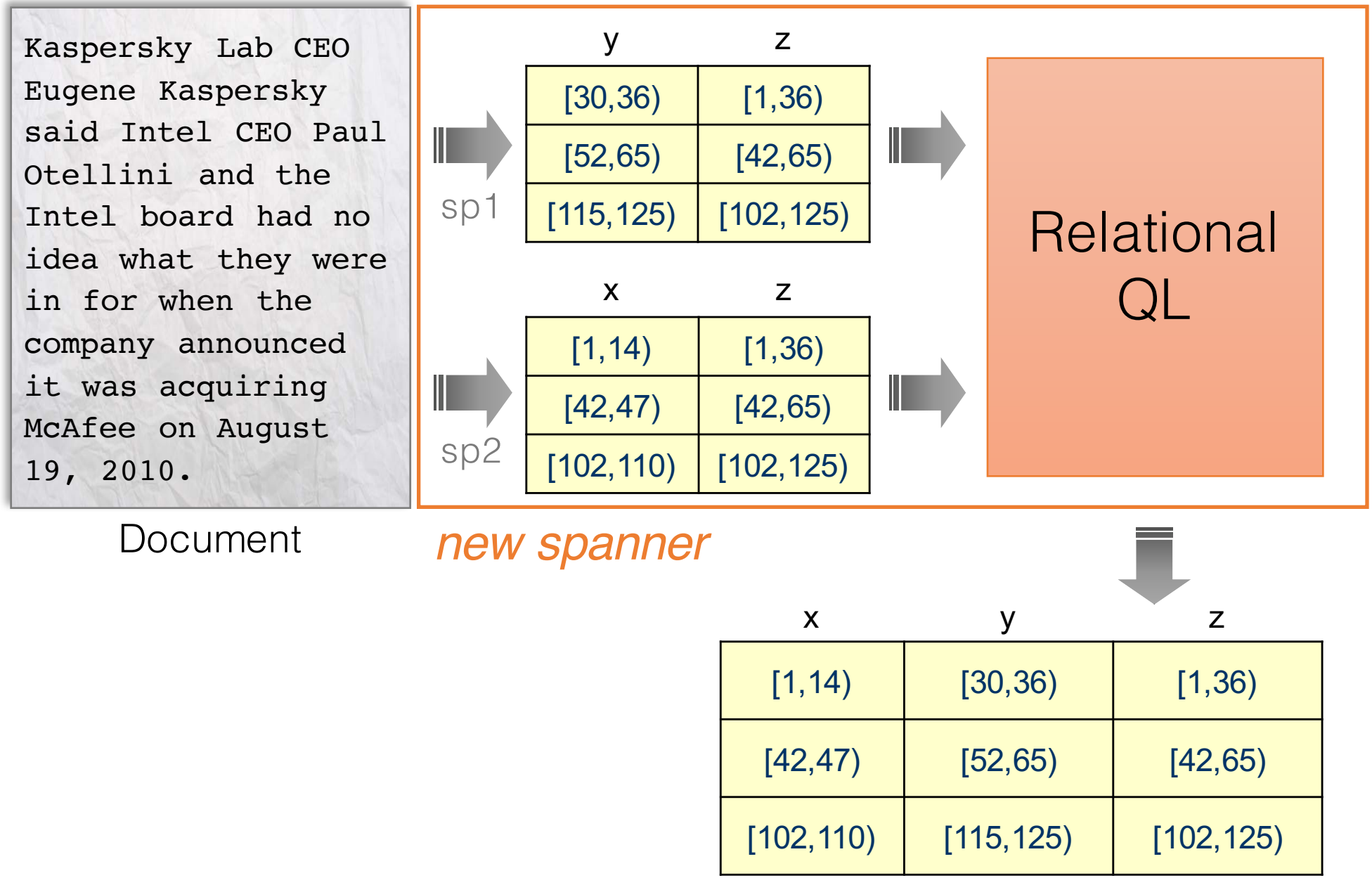


Document d

x	y	z
[1,14)	[30,36)	[1,36)
[42,47)	[52,65)	[42,65)
[102,110)	[115,125)	[102,125)

Relation over the spans of d

Spanners as Queries



What expressive power does relational QL add?

We began with a basic setup:

- Basic extraction by REGEX formulas
- Relational Algebra (RA)

Spanners as Regex Formulas

- Regular expression with embedded variables

$$\gamma := \underbrace{\emptyset \mid \epsilon \mid \sigma \mid \gamma \vee \gamma \mid \gamma \cdot \gamma \mid \gamma^* \mid}_{\text{Ordinary regex}} \underbrace{x\{\gamma\}}_{\text{Span variable}}$$

- Examples:
  - .* x{\d\d\d\d} .*
  - .* in w{Alabama | Alaska | Arizona | …} .*
  - (.* z{[A-Z][a-z]*, y{[A-Z][a-z]*}} .*) | …

- Restriction: each "evaluation" (parse tree) assigns one span to each variable (see [Fagin+,JACM15])

Representation system for spanners

Spanners as Datalog w/ Regex

```
Token(x) :=   [ (ε | .*_) x{[a-zA-Z]+} ( ((,V_) .*) | ε) ]
State(x) :=   Token(x) , [.* x{Georgia|Virginia|Washington}.*]
Cap1st(x) :=  Token(x) , [.* x{[A-Z].*}.*]
CommaSp(x,y,z) := [.* z{x{.*} ,_ y{.*}}.*]
Loc(z) :=     CommaSp(x,y,z) , Cap1st(x) , State(y)
RETURN(x,z) :=  Cap1st(x) , [.*x{.*}_from_z{.*}.*] , Loc(z)
```

EDBs = Spanners

Query goal → **RETURN**(x,z)

Carter_from_Plains,_Georgia,_Washington

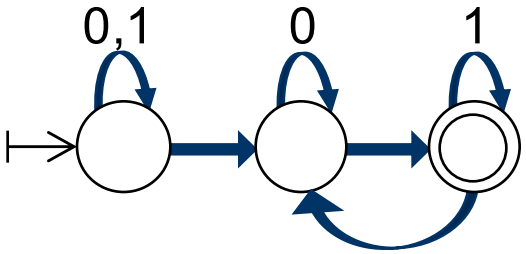
_from_Westmoreland,_Virginia

x	z
[1,7) *Carter*	[13,28) *Plains,_Georgia*
[30,40) *Washington*	[46,69) *Westmoreland,_Virginia*

Another representation system for spanners

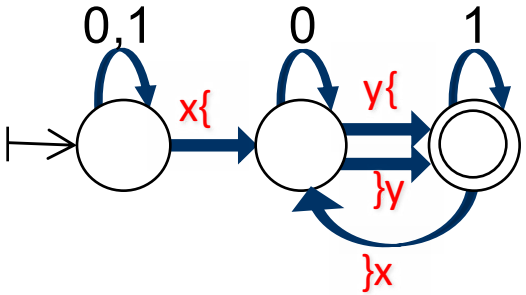
Spanners as Automata

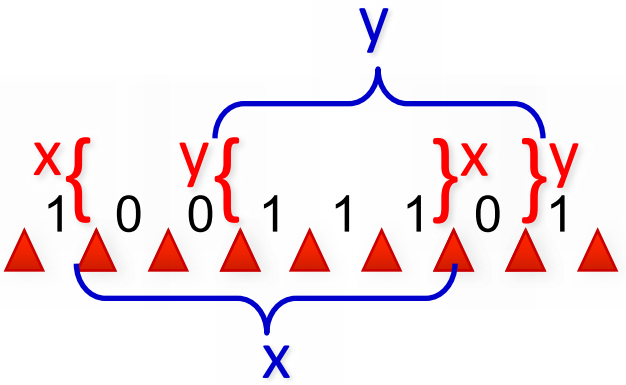
Ordinary NFA





Spanner Automaton

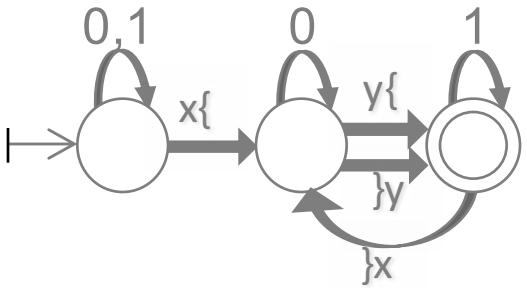




- In an *accepting run*, each variable opens and later closes exactly once
  ⇒ Each accepting run defines an assignment to the variables
- Nondeterministic ⇒ multiple runs ⇒ multiple tuples

Another representation system for spanners

Fundamental Result



Join	⋈
Union	∪
Product	×
Projection	π
Selection	ς
Difference	-

Spanners definable by
spanner automata

=

Spanners definable by
RA over regex formulas

=

Spanners definable by
NR-Datalog over regex formulas

```
Token(x)     :=  [ (ε | .*_) x{[a-zA-Z]+} ( ((,V_) .*) | ε) ]
State(x)     :=  Token(x) , [.* x{Georgia|Virginia|Washington}.*]
Cap1st(x)    :=  Token(x) , [.* x{[A-Z].*}.*]
CommaSp(x,y,z) :=  [.* z{x{.*} ,_ y{.*}}.*]
Loc(z)       :=  CommaSp(x,y,z) , Cap1st(x) , State(y)
RETURN(x,z)  :=  Cap1st(x) , [.*x{.*}_from_z{.*}.*] , Loc(z)
```

Consequences & Follow Ups

- Analysis of language extensions
  - Expressiveness, closure, difference, string operators [Fagin+, PODS14, JACM15]
- Principles of declarative cleaning in IE
  - [Fagin+, PODS14, TODS16]

Pain point!



Next topic

- Complexity analysis
  - [Freydenberger & Holldack, ICDT16, ICDT17]
- Uniform structured/unstructured DB
  - [Nahshon, Peterfreund, Vansummeren, WebDB16]

Outline

- Enterprise Search
- Information Extraction
- Prioritized Repairing

Cleaning IE Inconsistencies

- Extractors may produce inconsistent results
  - Data artifacts
  - Developer limitations





- Rather than repairing the existing extractors, common practice is to **clean** (intermediate) results
  - GATE/JAPE "controls" [Cunningham02]
  - POSIX regex disambiguation [Fowler03]
  - SystemT "consolidators" [Chiticariu+10]
  - Implicit in other rule systems, e.g., WHISK [Soderland99]

Implementation in IBM SystemT

```
create view Caps as
extract regex /[A-Z](\w|-)+/ on D.text as name from Document D;

create view Last as
extract dictionary LastGaz on D.text as name from Document D;

create view CapsLast as
select  CombineSpans(C.name, L.name) as name
from    Caps C, Last L
where   FollowsTok(C.name, L.name, 0, 0);

...

create view PersonAll as
    (select R.name from FirstLast R) union all ...
                    ... union all (select R.name from CapsLast R);

create view Person as select * from PersonAll R
consolidate on R.name using 'ContainedWithin';

output view Person;
```

Cleaning

[Chiticariu, Krishnamurthy, Li, Raghavan, Reiss, Vaithyanathan, ACL 2010]

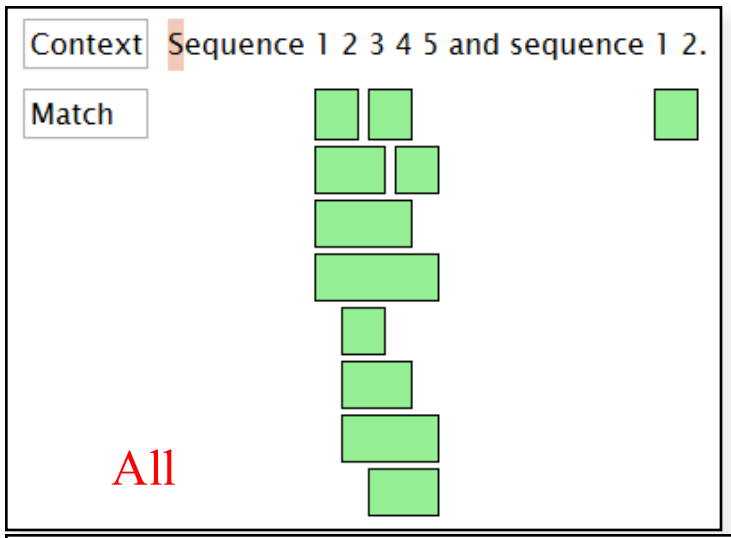
Five GATE/JAPE Controls

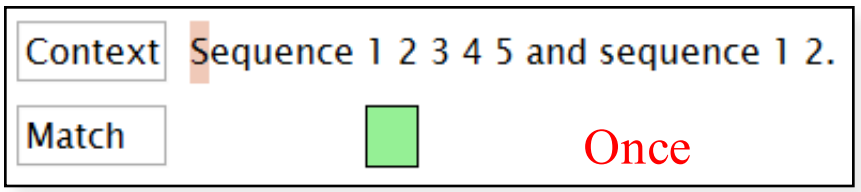
Sequence 12345 and sequence 12.

Document

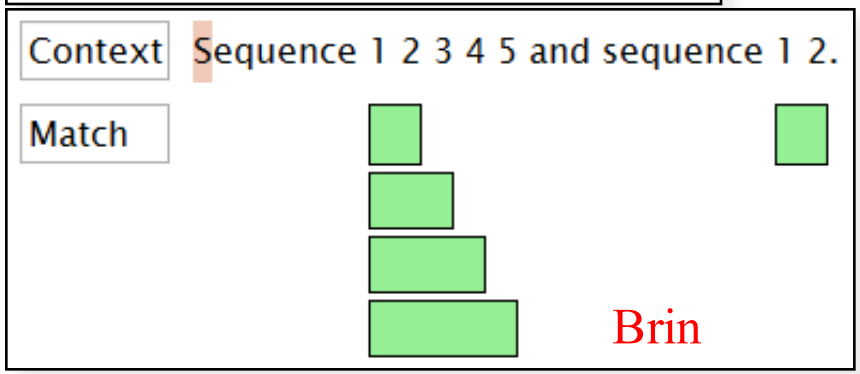
.* x{\d\d+} .*

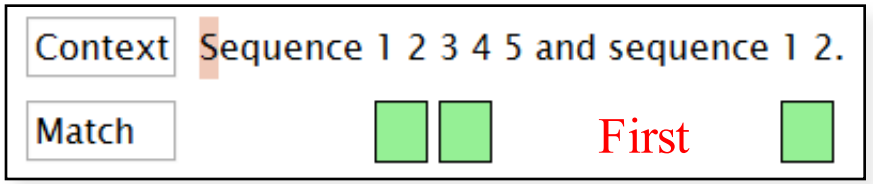
Spanner

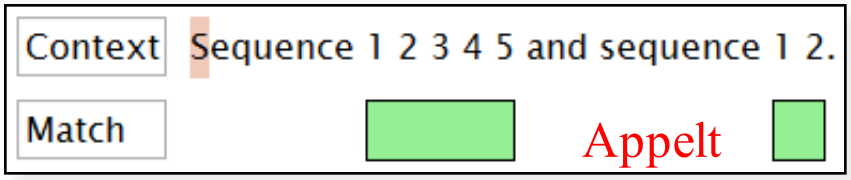




Screenshots from GATE UI







The University Of Sheffield.

GATE general architecture for text engineering

Declarative Cleaning

- Problem: existing policies are ad-hoc; *how to expose a language for user declaration?*
- We proposed a framework for **declarative cleaning** in IE [PODS14,TODS16]
- Can state rules like:

"denial constraints"

```
x and y are overlapping spans      →   not [ Person(x) & Location(y) ]

x and y separated by "and|or|,"    →   not [ Person(x) & Location(y) ]
```

```
y strictly contains x   →   Prefer Person(y)  to Person(x)

true                    →   Prefer Location(y)  to Person(x)
```

"priority relation"

Research Outcomes

- Framework based on:
  - Consistent query answering [Arenas+99]
  - Prioritized database repairs [Staworko+12]
- The framework captures, unifies, generalizes the policies of SystemT, GATE, POSIX, ...
- In addition, studied:
  - *When do the rules make sense?*
  - *When are the rules unambiguous?*
  - *Do cleaning rules add expressive power?*

Static analysis: quickly becomes **undecidable**

Prioritized Repairing

- We are given an **inconsistent database**, and a **preference relation** among tuples
  - Reliability, timestamps, semantics (divorced > single), ...
- Wish to lift preferences from **tuples** to **repairs**
  - Repair = maximal consistent subset of the database
- Several lifting alternatives [Staworko+12]
- We investigated complexity aspects:
  - **Repair checking:** *Is a given repair optimal?*
    - [Fagin, K, Kolaitis, PODS15]
  - **Categoricity:** *Is repairing ambiguous?*
    - [K, Livshits, Peterfreund, ICDT17]

Concluding Remarks

- Described 3 lines of research with Ron @Almaden
  - Enterprise search via *search database systems*
  - Foundations of IE via *document spanners*
  - Declarative cleaning in IE via *prioritized repairing*
- Current effort: stronger document spanners; uniform structured/unstructured; further prioritized repairing; …
- Takeaway: Again and again, "annoying details" led to fruitful fundamental research!

