Fundamentals

08/2021

Some Introductory Readings:

Foundations of Modern Query Languages for Graph Databases[1]

RENZO ANGLES, Universidad de Talca & Center for Semantic Web Research
MARCELO ARENAS, Pontificia Universidad Católica de Chile & Center for Semantic Web Research
PABLO BARCELÓ, DCC, Universidad de Chile & Center for Semantic Web Research
AIDAN HOGAN, DCC, Universidad de Chile & Center for Semantic Web Research
JUAN REUTTER, Pontificia Universidad Católica de Chile & Center for Semantic Web Research
DOMAGOJ VRGOČ, Pontificia Universidad Católica de Chile & Center for Semantic Web Research

We survey foundational features underlying modern graph query languages. We first discuss two popular graph data models: *edge-labelled graphs*, where nodes are connected by directed, labelled edges; and *property graphs*, where nodes and edges can further have attributes. Next we discuss the two most fundamental graph querying functionalities: *graph patterns* and *navigational expressions*. We start with graph patterns, in which a graph-structured query is matched against the data. Thereafter we discuss navigational expres-

Abrir en...

Querying Graph Databases

Pablo Barceló
Dept. of Computer Science, University of Chile
pbarcelo@dcc.uchile.cl

ABSTRACT

Graph databases have gained renewed interest in the last years, due to their applications in areas such as the Semantic Web and Social Networks Analysis. We study the problem of querying graph databases, and answering graph queries efficiently. We focus on two main properties of queries: their expressive power and their complexity. Expressive power is understood as the ability of a query language to express queries that are relevant in graph scenarios. Complexity refers to the worst-case asymptotic behavior of the time necessary to evaluate queries. We look at the traditional notion of graph queries as well as graph analytics, which are more complex queries that combine graph traversal and computation. Our goal is to understand what are the limits and what is the price to pay for querying graph databases, and to point to some of the most important results and techniques that have been proposed to deal with this price.

1. INTRODUCTION

Graph databases have gained renewed interest in the last years, due to their applications in areas such as the Semantic Web and Social Networks Analysis. We study the problem of querying graph databases, and answering graph queries efficiently. We focus on two main properties of queries: their expressive power and their complexity. Expressive power is understood as the ability of a query language to express queries that are relevant in graph scenarios. Complexity refers to the worst-case asymptotic behavior of the time necessary to evaluate queries. We look at the traditional notion of graph queries as well as graph analytics, which are more complex queries that combine graph traversal and computation. Our goal is to understand what are the limits and what is the price to pay for querying graph databases, and to point to some of the most important results and techniques that have been proposed to deal with this price.

1. INTRODUCTION

Graph databases have gained renewed interest in the last years, due to their applications in areas such as the Semantic Web and Social Networks Analysis. We study the problem of querying graph databases, and answering graph queries efficiently. We focus on two main properties of queries: their expressive power and their complexity. Expressive power is understood as the ability of a query language to express queries that are relevant in graph scenarios. Complexity refers to the worst-case asymptotic behavior of the time necessary to evaluate queries. We look at the traditional notion of graph queries as well as graph analytics, which are more complex queries that combine graph traversal and computation. Our goal is to understand what are the limits and what is the price to pay for querying graph databases, and to point to some of the most important results and techniques that have been proposed to deal with this price.

1. INTRODUCTION

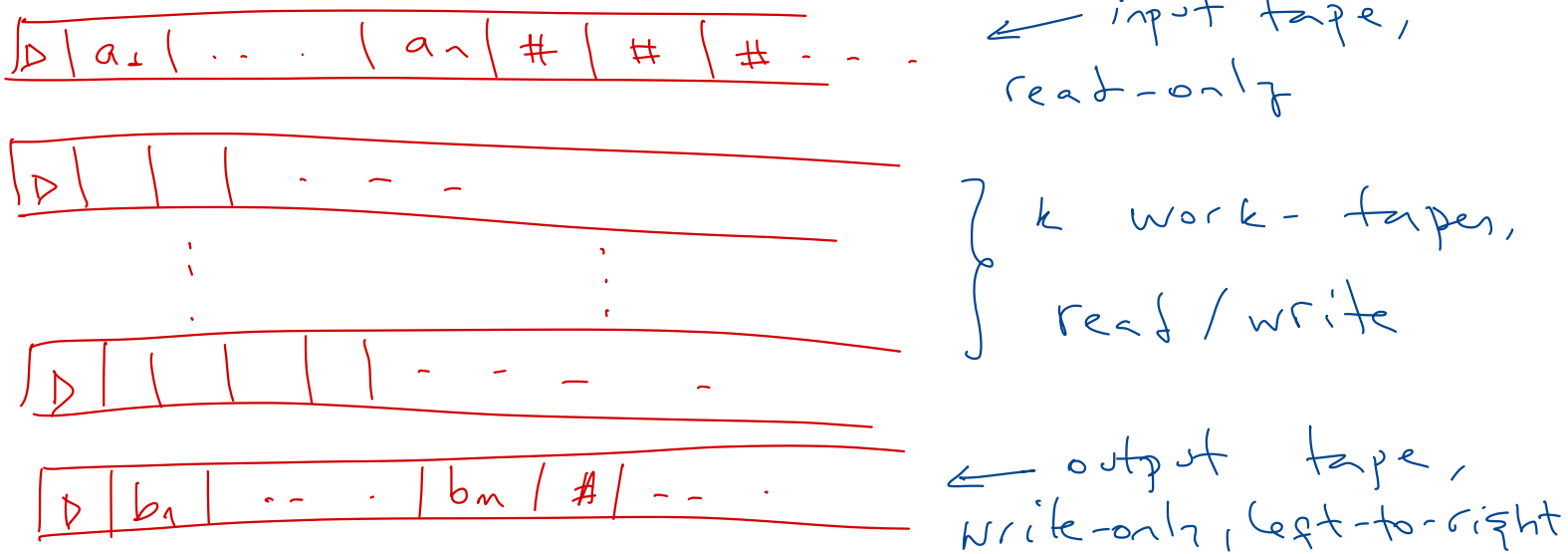
Graph databases have gained renewed interest in the last years, due to their applications in areas such as the Semantic Web and Social Networks Analysis. We study the problem of querying graph databases, and answering graph queries efficiently. We focus on two main properties of queries: their expressive power and their complexity. Expressive power is understood as the ability of a query language to express queries that are relevant in graph scenarios. Complexity refers to the worst-case asymptotic behavior of the time necessary to evaluate queries. We look at the traditional notion of graph queries as well as graph analytics, which are more complex queries that combine graph traversal and computation. Our goal is to understand what are the limits and what is the price to pay for querying graph databases, and to point to some of the most important results and techniques that have been proposed to deal with this price.

1. INTRODUCTION

Graph databases have gained renewed interest in the last years, due to their applications in areas such as the Semantic Web and Social Networks Analysis. We study the problem of querying graph databases, and answering graph queries efficiently. We focus on two main properties of queries: their expressive power and their complexity. Expressive power is understood as the ability of a query language to express queries that are relevant in graph scenarios. Complexity refers to the worst-case asymptotic behavior of the time necessary to evaluate queries. We look at the traditional notion of graph queries as well as graph analytics, which are more complex queries that combine graph traversal and computation. Our goal is to understand what are the limits and what is the price to pay for querying graph databases, and to point to some of the most important results and techniques that have been proposed to deal with this price.

BASICS OF COMPLEXITY THEORY

- We assume familiarity with Turing machines (TMs).
- We further assume they work on a tape-setting of the form:



- A Turing machine M accepts a word $\bar{\omega}$, if there exists a computation of M on $\bar{\omega}$ that reaches distinguished state q_{acc}.
- A Turing machine M rejects a word $\bar{\omega}$, if every computation of M on $\bar{\omega}$ reaches distinguished state q_{rej}.

- A language $L \subseteq \Sigma^*$ is decidable, if there is a TM M such that:
  a) It accepts every word $w \in L$
  b) It rejects every word $w \notin L$

Proposition: Every decidable language can be decided by a deterministic TM.

Example: The following problem is undecidable:

> Given words $\bar{w}_1, \ldots, \bar{w}_n$ and $\bar{v}_1, \ldots, \bar{v}_n$ over alphabet Σ, is there a sequence $1 \leq i_1, \ldots, i_n \leq n$ such that $\bar{w}_{i_1} \ldots \bar{w}_{i_n} = \bar{v}_{i_1} \ldots \bar{v}_{i_n}$?

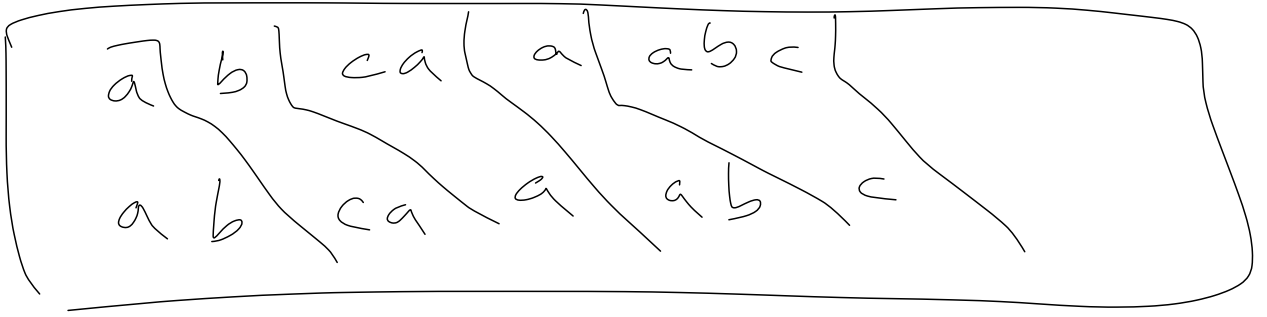
This is known as the Post Correspondence Problem (PCP).

Example :

$$\left[\frac{b}{ca}\right]_{V_1}^{w_1}, \left[\frac{a}{ab}\right]_{V_2}^{w_2}, \left[\frac{ca}{a}\right]_{V_3}^{w_3}, \left[\frac{abc}{c}\right]_{V_4}^{w_4}$$

Solution :

$$\left[\frac{a}{ab}\right]\left[\frac{b}{ca}\right]\left[\frac{ca}{a}\right]\left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right]$$



- Let $T : \mathbb{N} \to \mathbb{N}$. A TM M decides language $L \subseteq \Sigma^*$ in time $T(n)$, if:

1. Every computation of M on a word $\bar{w}$ takes at most $T(|\bar{w}|)$ steps.
2. M accepts every word in L.
3. M rejects each word <u>not</u> in L.

$DTIME(T(n)) = \{$ languages decided by deterministic TMs in time $O(T(n))$.

$NTIME(T(n)) = \{$ languages decided by non-deterministic TMs in time $O(T(n))$.

Important Classes:

$$P := \mathrm{DTIME}(n^{O(1)}) = \bigcup_{k \geq 0} \mathrm{DTIME}(n^k)$$

$$NP := \mathrm{NTIME}(n^{O(1)}) = \bigcup_{k \geq 0} \mathrm{NTIME}(n^k)$$

$$EXPTIME := \mathrm{DTIME}(2^{n^{O(1)}}) = \bigcup_{k \geq 0} \mathrm{DTIME}(2^{n^k})$$

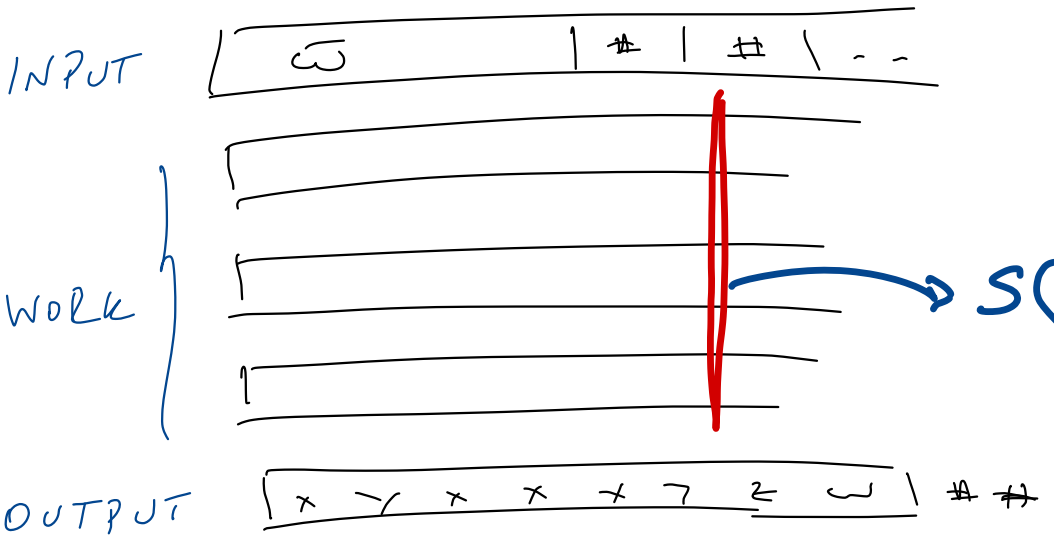
$$NEXPTIME := \mathrm{NTIME}(2^{n^{O(1)}}) = \bigcup_{k \geq 0} \mathrm{NTIME}(2^{n^k})$$

- P is what is accepted as "efficient" when dealing with data ("huge").
- When handling queries ("small"), we can afford extra cost.

- Let $S : \mathbb{N} \to \mathbb{N}$. A TM M decides language $L \subseteq \Sigma^*$ in space $S(n)$, if:

1. Every computation of M on a word $\bar{w}$ uses at most $S(|\bar{w}|)$ cells in the work-tapes.
2. M accepts every word in L.
3. M rejects each word not in L.

$\forall \bar{w} \in \Sigma^*$:



NO COMPUTATION DOES BEYOND THIS.

$DSPACE(S(n)) = \{$ languages decided by deterministic TMs in space $O(S(n))$.

$NSPACE(S(n)) = \{$ languages decided by non-deterministic TMs in space $O(S(n))$.

Important Classes:

$$\text{LOGSPACE} := \text{DSPACE}(\log n)$$

$$\text{NLOGSPACE} := \text{NSPACE}(\log n)$$

$$\text{PSPACE} := \text{DSPACE}(n^{O(1)}) = \bigcup_{k \geq 0} \text{DSPACE}(n^k)$$

$$\text{NPSPACE} := \text{NSPACE}(n^{O(1)}) = \bigcup_{k \geq 0} \text{NSPACE}(n^k).$$

$$\text{EXPSPACE} := \text{DSPACE}(2^{n^{O(1)}}) = \bigcup_{k \geq 0} \text{DSPACE}(2^{n^k})$$

$$\text{NEXPSPACE} := \text{NSPACE}(2^{n^{O(1)}}) = \bigcup_{k \geq 0} \text{NSPACE}(2^{n^k})$$

Savitch's Theorem: For any (well-behaved) function $S: \mathbb{N} \to \mathbb{N}$ with $S(n) \geq \log n$:

$$NSPACE(S(n)) \subseteq DSPACE(S^2(n))$$

Corollary:

- $NPSPACE = PSPACE$

Observation: This does <u>not</u> imply:

$$LOGSPACE = NLOGSPACE.$$

PSPACE = NPSPACE

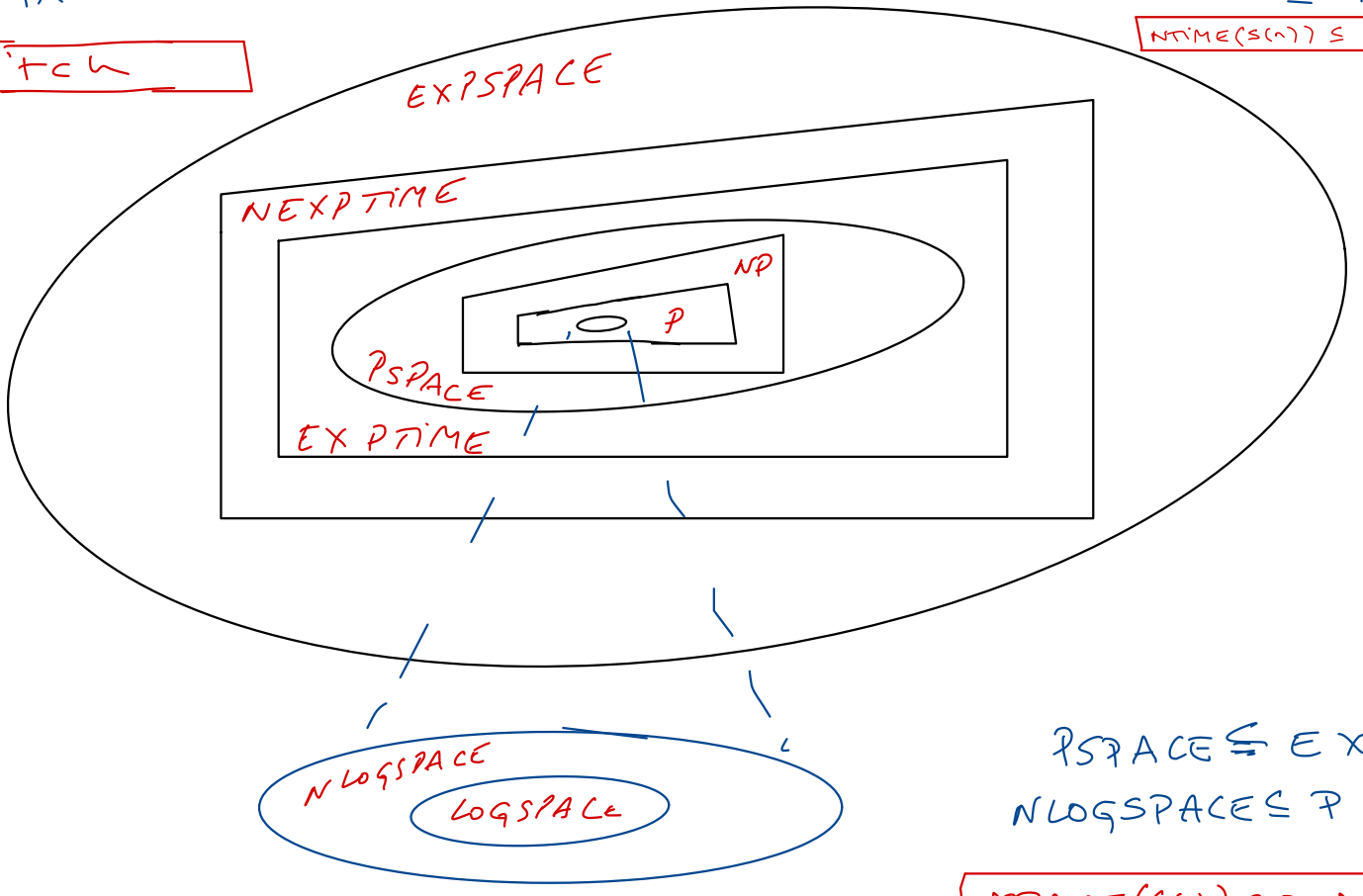
EXPSPACE = NEXPSPACE

NEXPTIME ⊆ EXPSPACE

NP ⊆ PSPACE

NTIME(S(n)) ⊆ SPACE(S(n))

Savitch



PSPACE ⊊ EXPTIME

NLOGSPACE ⊆ P

NSPACE(S(n)) ⊆ DTIME($2^{O(S(n))}$)

What do we know?

$P \subsetneq EXPTIME$ $\quad LOGSPACE \subsetneq PSPACE$

$\subsetneq 2EXPTIME$ $\quad \subsetneq EXPSPACE$

$\vdots$ $\quad \vdots$

What we do <u>not</u> know?

$P \subsetneq NP$?

$LOGSPACE \subsetneq NLOGSPACE$?

$NLOGSPACE \subsetneq P$?

$P \subsetneq PSPACE$?

$PSPACE \subsetneq EXPTIME$?

Complete problems are those that are "canonical" for a class $\mathcal{C}$, i.e. If they were solved in a "lower" complexity class $\mathcal{C}' \Rightarrow \mathcal{C} = \mathcal{C}'$.

Formally, $L \subseteq \Sigma^*$ is complete for one of the classes $\mathcal{C}$ seen before, if:

1. $L \in \mathcal{C}$
2. Every L' in $\mathcal{C}$ admits a (LOGSPACE)-reduction into L.

Important Complete problems:

NLOGSPACE: Given directed graph G, nodes s, t, is there directed path from s to t in G?

P: Given a boolean circuit B, and valuation η for the variables of B, does B evaluate to TRUE under η?

NP : Given propositional formula ϕ, is ϕ satisfiable?

PSPACE : Given quantified propositional formula

$\phi \equiv \exists \bar{x}_1 \forall \bar{x}_2 \ldots \exists \bar{x}_n \psi$, is ϕ valid?

BASICS OF AUTOMATA AND REGULAR LANGUAGES

Let Σ be a finite alphabet.

Regular expressions over Σ:

$$r, r' := \emptyset \mid \varepsilon \mid a \ (a \in \Sigma) \mid r + r' \mid r \cdot r' \mid r^*.$$

Example: The following are regular expressions over $\Sigma = \{a, b\}$:

$(aa)^*$, $(ab)^*$, $(a+b)^* b$.

Each regular expression r over Σ defines a regular language $L(r) \subseteq \Sigma^*$ as follows:

- $L(\emptyset) = \emptyset$
- $L(\varepsilon) = \{\varepsilon\}$ ← empty word
- $L(a) = \{a\}$
- $L(r + r') = L(r) \cup L(r')$
- $L(r \cdot r') = \{w \cdot w' \mid w \in L(r), w' \in L(r')\}$
- $L(r^*) = \bigcup_{i \geq 0} L(\underbrace{r \cdot r \cdot \ldots \cdot r}_{i \text{ times}})$

$$= \{w_1 \ldots w_n \mid n \geq 0, w_i \in L(r), \forall i \leq n\}$$

Example:

- $L((aa)^*) = \{\varepsilon, aa, aaaa, aaaaaa, \ldots\}$
  $= \{a^n \mid n \text{ is even}\}$
- $L((ab)^*) = \{\varepsilon, ab, abab, \ldots\}$
- $L((a+b)^* b) = \{\bar{w} \in \{a,b\}^* \mid$ last symbol in $\bar{w}$ is $b\}$.

Proposition: The following language is not regular:

$$\{a^n b^n \mid n \geq 0\}$$

That is, for no regular expression r we have that $\mathcal{L}(r) = \{a^n b^n \mid n \geq 0\}$.

- Regular expressions provide a grammar for defining languages.
- But languages can also be defined procedurally:

Automata: An automata over Σ is a tuple $A = (Q, q_0, F, \delta)$ with:

1. Q a finite set of states.
2. $q_0 \in Q$ an initial state.
3. $F \subseteq Q$ a set of final states.
4. $\delta \subseteq Q \times \Sigma \times Q$ a transition relation.

- Given a word

$$\bar{w} = a_1 a_2 \dots a_n \quad (n \geq 0)$$

over Σ, a run of $\mathcal{A}$ on $\bar{w}$ is a sequence

$$q_0 q_1 \dots q_n$$

such that $(q_{i-1}, a_i, q_i) \in \delta$, $\forall i$.

- The run is accepting if $q_n \in F$.
- The word $\bar{w}$ is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on $\bar{w}$.

Observation:

- There might be multiple runs of an automata A on $\bar{w}$.
- If $\bar{w} = \varepsilon$, then A accepts $\bar{w}$ iff $q_0 \in F$.

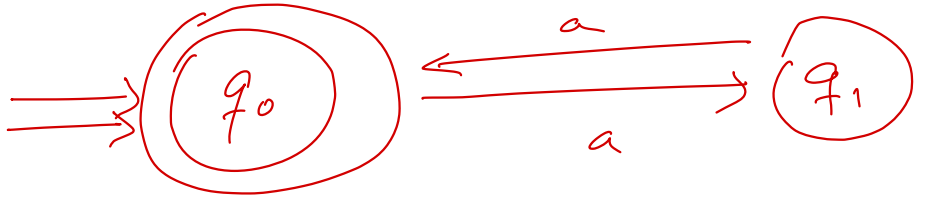
An automata A is deterministic if $(q, a, q'), (q, a, q'') \in \delta \Rightarrow q' = q''$, $\forall q, q', q'' \in Q$, $a \in \Sigma$.

∴ If A is deterministic $\Rightarrow$ only one run over every word.

The language defined by a is

$$\mathcal{L}(a) = \{\bar{\omega} \in \Sigma^* \mid a \text{ accepts } \bar{\omega}\}.$$

Example: The following automata defines the language $(aa)^*$.



← initial and final state.

Proposition. For every (non-deterministic) automata A, there exists a deterministic automata A' with $L(A) = L(A')$.

Theorem:

- For every regular expression r there is an automaton A with $\mathcal{L}(r) = \mathcal{L}(A)$.
- For every automaton A there is regular expression r with $\mathcal{L}(A) = \mathcal{L}(r)$.

Observation:

- The translation from a regular language into an equivalent automaton can be carried out in linear time.

- The opposite translation requires exponential time in the worst case.

Complexity of problems for automata:

- NLOGSPACE-complete: Given automaton A,
  - is $\mathcal{L}(A) \neq \emptyset$?
  - is $\mathcal{L}(A)$ finite?

- PSPACE-complete: Given A_1, A_2,
  - is $\mathcal{L}(A_1) \subsetneq \mathcal{L}(A_2)$?
  - is $\mathcal{L}(A_1) = \mathcal{L}(A_2)$?
  - is $\mathcal{L}(A_1) = \Sigma^*$?

Observation: Given automata $\mathcal{A}$, checking if $L(\mathcal{A}) \neq \emptyset$ can be carried out in linear time $O(|\mathcal{A}|)$.

Automata Intersection:

Given automata

- $A_1 = (Q_1, q_0^1, F_1, \delta_1)$
- $A_2 = (Q_2, q_0^2, F_2, \delta_2)$,

the automaton

$$A_1 \times A_2 = (Q_1 \times Q_2, (q_0^1, q_0^2), F_1 \times F_2,$$
$$\delta = \{[(q,r), a, (q',r')] \mid (q,a,q') \in \delta_1 \text{ and } (r,a,r') \in \delta_2\})$$

satisfies $L(A_1 \times A_2) = L(A_1) \cap L(A_2)$, and can be constructed in time $O(|A_1| \cdot |A_2|)$.

GRAPH DATABASE MODELS

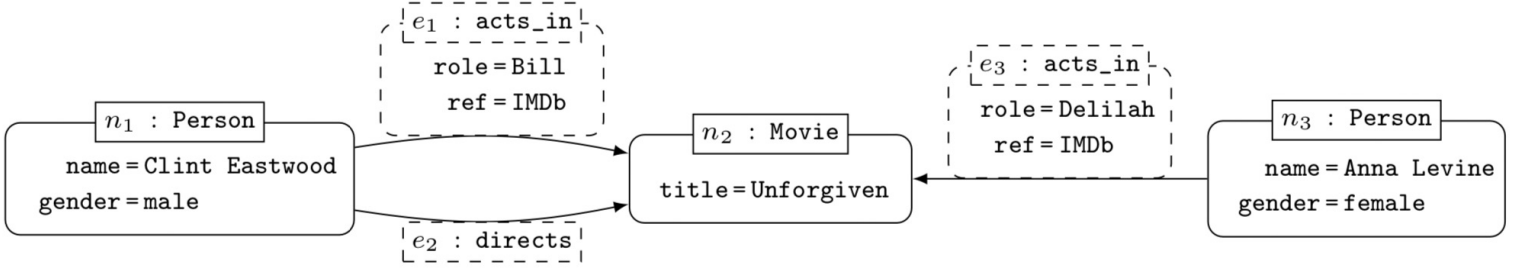
Graph databases are used to represent semi-structured data in certain domains where the "topology" of data is as important as data itself.

Example: Movie database, where

- Nodes are actors and movies
- Actors are linked to movies in which they act.
- Movies are linked if they have same director

Property graphs: The most general data model.

- Used by commercial graph DB engines.
- Allows to represent "topology" and "properties".



Formally: A property graph is a tuple $G = (V, E, \varphi, \lambda, \sigma)$, where:

- V finite set of node ids.
- E finite set of edge ids.
- $\varphi : E \to V \times V$ s.t. $\varphi(e) = (v_1, v_2)$ means e is edge from v_1 to v_2.
- $\lambda : (V \cup E) \to Lab$, where Lab is a set of "labels", and $\lambda(o)$ is label of o.
- $\sigma : (V \cup E) \times Prop \to Str$ is a partial function, for $Prop$ set of properties and Str set of strings.

$\sigma(o, p)$ is defined and takes value string w

$\Leftrightarrow$

property p is defined for object o and takes value p.

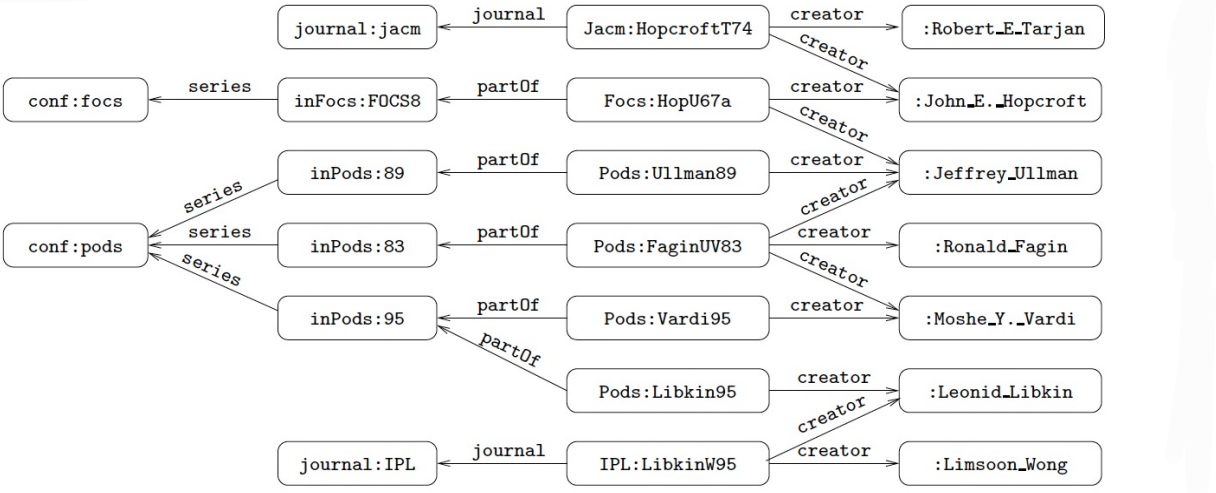
For theoretical exploration we often consider a simpler model of edge-labeled graphs.

Let Σ be a finite alphabet. A graph database over Σ is a tuple $G = (V, E)$ with:

- V a finite set of nodes
- $E \subseteq V \times \Sigma \times V$ a set of edges labeled in Σ.

We write $u \xrightarrow{a} v$ to denote that there is an edge from node u to node v labeled a.

Example:



DECISION PROBLEMS ON GRAPH DATABASES.

Problems investigated in graph databases are related to query languages, i.e., languages used to retrieve data from graph DBs.

Expressive power: Which properties of nodes/graphs can be expressed?

Query evaluation: Given a query q, a graph DB G, a tuple $\bar{t}$ of nodes, is $\bar{t}$ in $q(G)$, the "evaluation" of q over G?

Containment: Given queries q, q', is it the case that

$$q(G) \subseteq q'(G), \quad \forall G?$$

Complexity of query evaluation comes in two flavors:

1. Combined complexity: The query q is part of the input.
2. Data complexity: The query is fixed, i.e., its size $\|q\|$ is constant.

Data complexity approaches the fact that:
- D is "large"
- q is "small".