Path Querying on Graph Databases

Jelle Hellings
Hasselt University and transnational University of Limburg

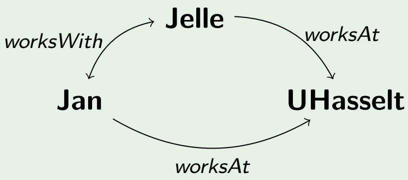
Overview

Overview

Graphs

- Pieces of data (nodes)
- Relations between the pieces of data (edges)

Example (Social networks)



Applications

- XML and RDF,
- Social networks,
- Transportation networks,
- The World Wide Web,
- ...

Graph Database: Google Maps

- Nodes: points of interest, addresses, ...
- Edges: road network
- Queries:

Example (Distance based query)

university close to <my address>
(answer: Universiteit Hasselt; 5.2 km)

Example (Route-planning query)

From: <my address>, to: <university>
(answer: options for university; followed by route)

Challenges

- Engineering: big data
  Storage, distributed processing, hardware failures, ...
- Conceptual: semantics and consistency
  Structured data (facebook) versus structured? data (the web)
- Conceptual: data querying
  Local/navigational based versus graph-wide path based
  - No widely used general purpose languages
  - Current practice: application specific languages

Challenges

- Engineering: big data
  Storage, distributed processing, hardware failures, ...
- Conceptual: semantics and consistency
  Structured data (facebook) versus structured? data (the web)
- Conceptual: data querying
  Local/navigational based versus graph-wide path based
  - No widely used general purpose languages
  - Current practice: application specific languages

Our focus

Path-based graph querying

Overview

Motivation

- Expressing graph-queries
- Properties of paths, walks, . . .

Route planning

We want to travel from *our office* to a *cafetaria* and from this *cafetaria* get back to the *office* using a *different route*

Graph querying

Graphs as traditional relations

$$\textbf{worksAt}(\textit{person}, \textit{company})$$

- Already deep knowledge of these systems
- First-order based query languages:

$$Q(n) := \exists m\, \textbf{worksWith}(n, m) \land \textbf{worksAt}(m, \texttt{UHasselt})$$

- Largely restricted to 'local' reasoning
  no paths, no or only limited reachability, ...

Higher-order logics

Monadic second-order logic

Extend first-order logic with quantification over sets

- Strong theoretical background
  - Sets with only nodes versus sets with nodes and edges
- Some graph problems are naturally expressible with sets:
  - Graph coloring, bipartite graph, ...

$$\begin{aligned}\exists S \exists T\, (\forall x\, (x \in S \vee x \in T) \wedge \\ (x \in S \implies x \notin T) \wedge (x \in T \implies x \notin S) \wedge \\ \forall y\, edge(x, y) \implies ((x \in S \wedge y \in T) \vee (y \in S \wedge x \in T)))\end{aligned}$$

- Paths non-straightforward: *y is reachable from x*

$$\forall S\, [(x \in S) \wedge \forall u \forall v\, (u \in S \wedge edge(u, v) \implies v \in S) \implies y \in S]$$

Conjunctive Regular Path Queries

Idea

- Query nodes based on labelling of paths between nodes
- Express labelling by a *regular expression*

Example

$$Q(a, b) := a \pi b, (\alpha\beta + \gamma\delta)^{+}(\pi)$$

$$\begin{array}{ccccccc}
n_1 & \xrightarrow{\alpha} & n_2 & \xrightarrow{\beta} & n_3 & \xrightarrow{\gamma} & n_4 \\
\delta\big\downarrow & & & & & & \big\downarrow\delta \\
n_5 & \xrightarrow{\gamma} & n_6 & \xrightarrow{\alpha} & n_7 & \xrightarrow{\beta} & n_8
\end{array}$$

Conjunctive Regular Path Queries

Idea

- Query nodes based on labelling of paths between nodes
- Express labelling by a *regular expression*

Example

$$Q(a, b) := a\pi b, (\alpha\beta + \gamma\delta)^{+}(\pi)$$

$$\begin{array}{ccccccc}
n_1 & \xrightarrow{\alpha} & n_2 & \xrightarrow{\beta} & n_3 & \xrightarrow{\gamma} & n_4 \\
\delta\downarrow & & & & & & \downarrow\delta \\
n_5 & \xrightarrow{\gamma} & n_6 & \xrightarrow{\alpha} & n_7 & \xrightarrow{\beta} & n_8
\end{array}$$

Conjunctive Regular Path Queries

Idea

- Query nodes based on labelling of paths between nodes
- Express labelling by a *regular expression*

Example

$$Q(a, b) := a\pi b, (\alpha\beta + \gamma\delta)^+(\pi)$$

$$\begin{array}{ccccccc}
n_1 & \xrightarrow{\alpha} & n_2 & \xrightarrow{\beta} & n_3 & \xrightarrow{\gamma} & n_4 \\
\delta\downarrow & & & & & & \downarrow\delta \\
n_5 & \xrightarrow{\gamma} & n_6 & \xrightarrow{\alpha} & n_7 & \xrightarrow{\beta} & n_8
\end{array}$$

Extended Conjunctive Regular Path Queries

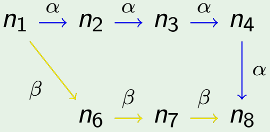
Idea

Comparing labelling of paths

- Regular expressions over n-tuples
- Use special symbol $\perp$ to specify end-of-path

Example

$$Q(a, b) := a\pi_1 b, a\pi_2 b, (\left[\begin{smallmatrix}\alpha \\ \beta\end{smallmatrix}\right]^+ \left[\begin{smallmatrix}\alpha \\ \perp\end{smallmatrix}\right])(\pi_1, \pi_2)$$

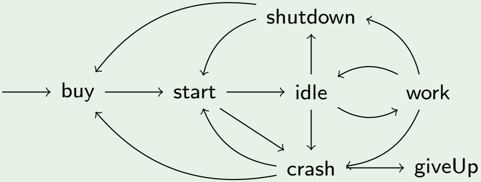


Computation tree logic*

Usage: verification of formal models

- Describe behaviour by a *transition system* (graph)
- Write propositions that should hold

Example

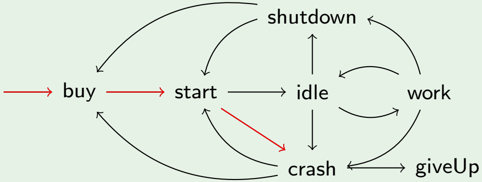


Computation tree logic*

Usage: verification of formal models

- Describe behaviour by a *transition system* (graph)
- Write propositions that should hold

Example



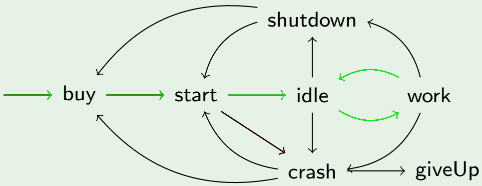
Machine never crashes: $\mathbf{A}\,\mathbf{G}\,\neg \mathit{crash}$

Computation tree logic*

Usage: verification of formal models

- Describe behaviour by a *transition system* (graph)
- Write propositions that should hold

Example



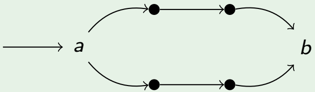
Machine can work without crashing: $\mathbf{E}\,\mathbf{G}\,\neg \mathit{crash}$

Hybrid CTL*

Idea

- CTL* has only implicit paths and nodes
- Add ability to *name* nodes in our formulae

Example



We can get from a to b in two different ways:

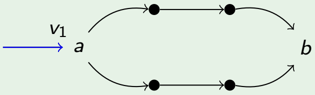
$$\mathbf{E} \downarrow_{v_1} \mathbf{E}\,\mathbf{F}(\downarrow_{v_2} \mathbf{E}\,\mathbf{X}\,\mathbf{F}(b \wedge \downarrow_{v_3} @_{v_1} \mathbf{E}(\neg v_2\, \mathbf{U}\, v_3)))$$

Hybrid CTL*

Idea

- CTL* has only implicit paths and nodes
- Add ability to *name* nodes in our formulae

Example



We can get from a to b in two different ways:

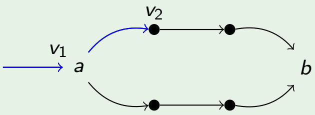
$$\mathbf{E} \downarrow_{v_1} \mathbf{E}\,\mathbf{F}(\downarrow_{v_2} \mathbf{E}\,\mathbf{X}\,\mathbf{F}(b \wedge \downarrow_{v_3} @_{v_1} \mathbf{E}(\neg v_2\, \mathbf{U}\, v_3)))$$

Hybrid CTL*

Idea

- CTL* has only implicit paths and nodes
- Add ability to *name* nodes in our formulae

Example



We can get from a to b in two different ways:

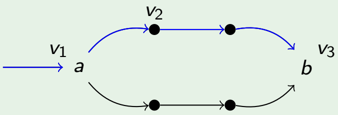
$$\mathbf{E} \downarrow_{v_1} \mathbf{E}\,\mathbf{F}(\downarrow_{v_2} \mathbf{E}\,\mathbf{X}\,\mathbf{F}(b \wedge \downarrow_{v_3} @_{v_1} \mathbf{E}(\neg v_2 \,\mathbf{U}\, v_3)))$$

Hybrid CTL*

Idea

- CTL* has only implicit paths and nodes
- Add ability to *name* nodes in our formulae

Example



We can get from a to b in two different ways:

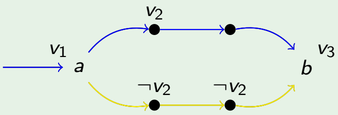
$$\mathbf{E} \downarrow_{v_1} \mathbf{E}\,\mathbf{F}(\downarrow_{v_2} \mathbf{E}\,\mathbf{X}\,\mathbf{F}(b \wedge \downarrow_{v_3} @_{v_1} \mathbf{E}(\neg v_2\, \mathbf{U}\, v_3)))$$

Hybrid CTL*

Idea

- CTL* has only implicit paths and nodes
- Add ability to *name* nodes in our formulae

Example



We can get from a to b in two different ways:

$$\mathbf{E}\downarrow_{v_1}\ \mathbf{E}\,\mathbf{F}(\downarrow_{v_2}\ \mathbf{E}\,\mathbf{X}\,\mathbf{F}(b\wedge\downarrow_{v_3}\ @_{v_1}\ \mathbf{E}(\neg v_2\,\mathbf{U}\,v_3)))$$

Overview

Walk Logic

Idea: extend first-order logic

- Add *walks*
- Add *positions on walks*
- Necessary operators to compare positions

Route planning

We want to travel from *our office* to a *cafetaria* (R) and from this *cafetaria* get back to the *office* using a *different route* (S)

$$\exists R \exists S \exists {t_1}^R \exists {t_2}^R \exists {u_1}^S \exists {u_2}^S \exists {u_3}^S$$
$$(\mathrm{office}(t_1) \land t_1 < t_2 \land \mathrm{cafeteria}(t_2) \land u_1 < u_3 < u_2$$
$$\land\, u_1 \sim t_2 \land u_2 \sim t_1 \land \forall {t_3}^R (t_1 < t_3 < t_2 \implies t_3 \not\sim u_3))$$

Definitions

Definition (Directed node-labeled graph)

A directed node-labeled graph is a triple $G = (N, E, l)$:

- N is a finite set of *nodes*
- $E \subseteq N \times N$ is the set of *edges*
- $l : N \to 2^{\mathcal{AP}}$ is a node-label function

Walk Logic

- Walk variables
- Position variables per walk variable

Atomic Formulae

$a(t)$	Node referred to by position variable t has labelling a
$t_1 \sim t_2$	Position variables t_1, t_2 refer to the same node
$t_1 < t_2$	Position variable t_1 comes before t_2 in walk W
	Position variables t_1 and t_2 *must* be of the same sort

φ, ψ are formulae

$\neg\varphi, \varphi \vee \psi$	Negation and disjunction
$\exists W\, \varphi$	Quantification over *walks*
$\exists t^W\, \varphi$	Quantification over *positions on walks*

Semantics: 'walks'?

Definition

Infinite walk A finite or infinite sequence $v_1 \ldots$ of nodes such that $(v_i, v_{i+1}) \in E$ for each $1 \leq i \leq |v_1 \ldots|$

Walk A nonempty finite sequence $v_1 \ldots v_n$ of nodes such that $(v_i, v_{i+1}) \in E$ for each $1 \leq i \leq n$

Trail A *walk* without edge repetition

Path A *walk* without node repetition

Semantics: 'walks'?

Definition

Infinite walk: A finite or infinite sequence $v_1 \ldots$ of nodes such that $(v_i, v_{i+1}) \in E$ for each $1 \leq i \leq |v_1 \ldots|$

Walk: A nonempty finite sequence $v_1 \ldots v_n$ of nodes such that $(v_i, v_{i+1}) \in E$ for each $1 \leq i \leq n$

Trail: A *walk* without edge repetition

Path: A *walk* without node repetition

- CTL* and Hybrid CTL*: primarily infinite walks
- CRPQs: primarily walks

Semantics: expressive power

Hierarchy of expressive power

Infinite walk A walk W is finite:

$$\exists t^W \neg \exists u^W \; t < u$$

Walk A *walk* is a *trail* (informally):

$$\forall t^W \forall u^W \left(t \sim u \land t_{+1} \sim u_{+1}\right) \implies t = u$$

Trail A *trail* W is a *path* (informally):

$$\forall t^W \forall u^W \; t \sim u \implies t = u$$

Path Logic $\preceq$ *Trail Logic* $\preceq$ *Walk Logic* $\preceq$ *Infinite Walk Logic*

Walk-based Graph Properties

Example (Hamiltonian Path (in Path Logic))

$$\exists P \forall Q \forall t^Q \exists u^P \, (t \sim u)$$

Example (Eulerian Trail (in Trail Logic))

$$\exists T \forall Q \forall t^Q \exists u^T \, (t \sim u) \wedge (t_{+1} \sim u_{+1})$$

Example (Strongly Connected)

$$\forall P \forall Q \forall t^P \forall u^Q \exists R \exists v^R \exists w^R \, (v < w \wedge t \sim v \wedge u \sim w)$$

Properties on undirected graphs

Theorem

Weakly Connected is not expressible on directed graphs

Proof.

$$n_1 \longleftarrow n_2 \longrightarrow n_3 \longleftarrow n_4 \longrightarrow n_5 \longleftarrow n_6$$

All walks contain at most 2 nodes: *reduce to first-order logic* □

- Direction matters!
- On undirected graphs:
  Weakly Connected same way as strongly connected
  Planar Graph using Kuratowski's Theorem

Overview

MSO(nodes, edges) and paths

Observations

- Path: sequence of connected edges
- No node repetition: nodes and positions coincide
- Node a before node b on path P if and only if
  Node b is reachable from a using the edges in P

Theorem

Path Logic $\prec$ *MSO(nodes, edges)*

Set-based Graph Properties

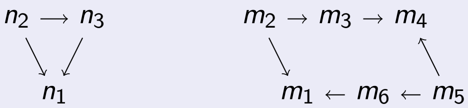
Theorem

Bipartite graph is not expressible on directed graphs

Lemma (Dénes Kőnig)

A graph is bipartite iff it does not contain an odd cycle

Proof.



All walks contain at most 3 nodes: *reduce to first-order logic* □

- MSO(nodes) *can* express bipartite graph
- Is Walk Logic strictly subsumed by MSO?

Eulerian Trail

Theorem

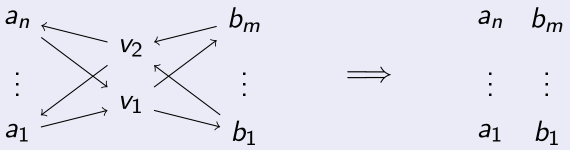
MSO(nodes, edges) cannot express Eulerian Trail

Lemma (well known result)

MSO cannot distinguish sets with i from sets with j elements

Proof.

For MSO: existence of Eulerian Trail in the graph



Reduces to sets A and B having the equal number of elements □

Relations with FO and MSO

- We have *FO* $\prec$ *Path Logic* $\prec$ *MSO(nodes, edges)*
- *Trail Logic*, *Walk Logic*, and *Infinite Walk Logic* are incomparable with *MSO(nodes)* and *MSO(nodes, edges)*

Lemma (Courcelle and Engelfriet)

MSO(nodes) cannot express Hamiltonian Path

- *Path Logic* and *MSO(nodes)* are incomparable

Path Logic $\prec$ *Trail Logic* $\prec$ [1]*Walk Logic* $\preceq$ *Infinite Walk Logic*

[1]The proof for *Trail Logic* $\prec$ *Walk Logic* is omitted but is similar to the proof of *Path Logic* $\prec$ *Trail Logic*

Overview

Review: Hybrid CTL*

Definition

Let a be an atomic proposition, x a node variable, φ_1 and φ_2 node formulas, ψ_1 and ψ_2 path formulas

Node formulas

$$\varphi ::= a \mid x \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \downarrow_x \varphi_1 \mid @_x \varphi_1 \mid \mathbf{E}\, \psi$$

Path formulas

$$\psi ::= \varphi \mid \neg\psi_1 \mid \psi_1 \vee \psi_1 \mid \mathbf{X}\, \psi_1 \mid \psi_1 \, \mathbf{U} \, \psi_2$$

Translating Hybrid CTL* to Walk Logic

Idea

Node formulas Properties of single node:
translate to properties of single position

Hybrid extensions Named nodes:
translate to named position variables

Path formulas Properties on a single path with forward navigation:
translate to walk variable; keep track of current position using position variables and $<$

Translating Hybrid CTL* to Walk Logic

Idea

Node formulas Properties of single node:
translate to properties of single position

Hybrid extensions Named nodes:
translate to named position variables

Path formulas Properties on a single path with forward navigation:
translate to walk variable; keep track of current position using position variables and $<$

$$\mathit{CTL}^* \prec \mathit{Hybrid\ CTL}^* \preceq \mathit{Infinite\ Walk\ Logic}$$

*Hybrid CTL** $\prec$ *Infinite Walk Logic*?

Theorem

*Hybrid CTL** $\prec$ *Infinite Walk Logic*

Proof.

- *CTL** $\prec$ *Infinite Walk Logic* as CTL* is *invariant under bisimulation*
- *Hybrid CTL** $\prec$ *Infinite Walk Logic* as Hybrid CTL* is *invariant under generated submodels*

□

Hybrid $CTL^* \prec$ *Infinite Walk Logic*?

Theorem

Hybrid $CTL^* \prec$ *Infinite Walk Logic*

Proof.

- $CTL^* \prec$ *Infinite Walk Logic* as CTL^* is *invariant under bisimulation*
- *Hybrid* $CTL^* \prec$ *Infinite Walk Logic* as Hybrid CTL^* is *invariant under generated submodels*

□

$CTL^* \prec$ *Hybrid* $CTL^* \prec$ *Infinite Walk Logic*

*Hybrid CTL** $\prec$ *Infinite Walk Logic?*

Theorem

*Hybrid CTL** $\prec$ *Infinite Walk Logic*

Proof.

- *CTL** $\prec$ *Infinite Walk Logic* as CTL* is *invariant under bisimulation*
- *Hybrid CTL** $\prec$ *Infinite Walk Logic* as Hybrid CTL* is *invariant under generated submodels*

□

*CTL** $\prec$ *Hybrid CTL** $\prec$ *Infinite Walk Logic*

Walk Logic $\preceq$ *Infinite Walk Logic*

Overview

CRPQs versus Walk Logics

Different languages!

Focus on path labelling versus focus on path-structure of graphs

- All CRPQs are incomparable with all Walk Logics.

CRPQs versus Walk Logics

Different languages!

Focus on path labelling versus focus on path-structure of graphs

- All CRPQs are incomparable with all Walk Logics.

- Similar semantically questions
- Similar proof techniques

Some results

- Hamiltonian path cannot be expressed
- Eulerian trail cannot be expressed

Some results

- Hamiltonian path cannot be expressed
- Eulerian trail cannot be expressed

Paths versus Walks

CRPQ with paths can express queries not expressible in the strongest language with Walks!

Overview

Graph Databases

Motivation

Walk Logic

Relations with FO and MSO

Relations with CTL* and Hybrid CTL*

Conjunctive Regular Path Queries

Open Problems and Conclusion

Open Problems

- Walk Logic versus Infinite Walk Logic:
  - Infinite walks are the standard in verification logics
  - Can we express the verification logics in Walk Logic?
  - Also interesting: finite CTL$*$ versus infinite CTL*
- Complexity bounds on model checking for WL:
  - WL model checking is decidable
  - Current approach has horrible complexity

Conclusion

- General walk-based reasoning on graphs
- Relates to practical graph languages
- Framework for studying expressivity