Syntax and Semantics of Parameterized Regular Expression

Heyang Li

August 2024

Property Graph Model in Millennium DB

Formally, assume we have a universe of objects $\texttt{Obj}$, a set of labels L, and a set of attributes $Attr$, and a set of values Val. We summarize the data model of Millennium DB from [1] as following:

Definition 1 (Domain Graph)**.** *A domain graph $G = (O, \gamma)$ consists of $O \subseteq \texttt{Obj}$ is a set of objects, and $\gamma : O \to O \times O \times O$*

O is the set of objects in the graph database, and the relation γ models edges between objects. $\gamma(e) = (n_1, t, n_2)$ states that the edge (n_1, t, n_2) has id e, type t, source node n_1 and target node n_2. The id e is generated by the database.

Assume we have a set of labels L, a set of attributes $Attr$ and a set of values Val, we can define property domain graph as follows:

Definition 2 (Property Domain Graph)**.** *A property domain graph is defined as a tuple $G = (O, \gamma, \texttt{lab}, \texttt{prop})$,5 where:*

- *(O, γ) is a domain graph.*
- *$\texttt{lab} : O \to 2^L$ is a function assigning a finite set of labels to an object.*
- *$\texttt{attr} : O \times Attr \to Val$ is a partial function assigning a value to a certain property of an object.*

Assume we have the following domain property graph

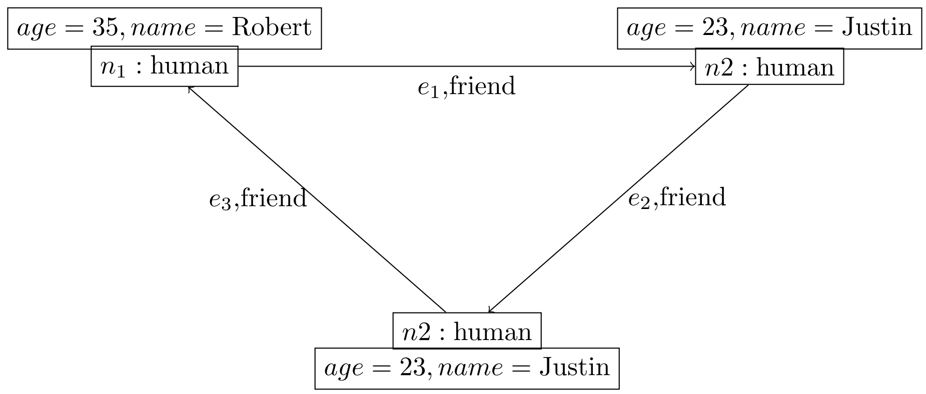

Figure 1: An example of domain property graph

There are three edges: $\gamma(e_1) = (n_1, friend, n_2)$, $\gamma(e_2) = (n_2, friend, n_3)$,$\gamma(e_3) = (n_3, friend, n_1)$.

We assign each node a label 'human', i.e. $\texttt{label}(n_1) = \text{human}$, $\texttt{label}(n_2) = \text{human}$, $\texttt{label}(n_3) = \text{human}$.

The Syntax of Parameterized Regular Expression

Here is proposed syntax for parameterized regular expression. Assume we have global parameters $\mathbb{P}$, a set of labels L and a set of attributes $Attr$. We formalize the grammar of parameterized regular expression as following:

$$
\begin{array}{lcll}
E & ::= & (t, \phi) & \\
 & | & \hat{} E & \\
 & | & E_1 / E_2 & \\
 & | & E_1 \mid E_2 & \\
 & | & E^* & \\
 & | & E^+ & \\
 & | & E^? & \\
 & | & (E) & \\
t & ::= & T \in \texttt{TYPE} & \textit{TYPE} \subset O \\
 & | & l \in L & \\
\phi & ::= & \phi \wedge \phi & \\
 & | & x = c_{str} & x \in Attr \textit{ and c is a string constant} \\
 & | & t_{ar} \sim t_{ar} & \sim \in \{>, <, \leq, \geq, \neq, =\} \\
t_a r & ::= & \Sigma_i a_i p_i & a_i \in \mathbb{R} \\
p & ::= & attr & attr \in Attr, \textit{ i.e. attributes of objects} \\
 & | & ?p & ?p \in \mathbb{P}, \textit{i.e. global parameters}
\end{array}
$$

The precedence of the syntax in regular expression is:

- Path atoms (t, ϕ)

- Groups in ()
- Unary operators *, ? and +
- Unary ˆ inverse links
- Binary operator /
- Binary operator |

The Semantics of Parameterized Regex on Property Graph

Projection functions Given a parameterized regex $E((t_1, \phi_1), \ldots, (t_n, \phi_n))$, where $(t_1, \phi_1), \ldots, (t_n, \phi_n)$ are atomic expressions. Here we define two projection functions, fst and snd.

$$
\begin{aligned}
fst(E((t_1, \phi_1), \ldots, (t_n, \phi_n))) &= E(t_1, \ldots, t_n) \\
fst((t, \phi)) &= t \\
fst(\hat{}E) &= \hat{}(fst(E)) \\
fst(E)^* &= (fst(E))^* \\
fst(E)^+ &= (fst(E))^+ \\
fst(E_1/E_2) &= (fst(E_1)/fst(E_2)) \\
fst(E_1 \mid E_2) &= (fst(E_1) \mid fst(E_2)) \\
fst((E)) &= ((fst(E)))
\end{aligned}
$$

Intuitively, fst operation picks the type/label domain of each atomic from a parameterized regular expression. For example, if we have a parameterized regular expression $e = (n_1, ?p > age)^*/(t_1, name = \texttt{Swen})$, then we have $fst(e) = n_1^*/t_1$.

$$
\begin{aligned}
snd(E((t_1, \phi_1), \ldots, (t_n, \phi_n))) &= E(\phi_1, \ldots, \phi_n) \\
snd((t, \phi)) &= \phi \\
snd(\hat{}E) &= \hat{}(snd(E)) \\
snd(E)^* &= (snd(E))^* \\
snd(E)^+ &= (snd(E))^+ \\
snd(E_1/E_2) &= (snd(E_1)/snd(E_2)) \\
snd(E_1 \mid E_2) &= (snd(E_1) \mid snd(E_2)) \\
snd((E)) &= ((snd(E)))
\end{aligned}
$$

snd operation picks the formula domain of each atomic from a parameterized regular expression. For the above example, we have $snd(e) = (?p > age)^*/(name = \texttt{Swen})$.

Path Assume we have a domain property graph $G = (O, \gamma, \texttt{label}, \texttt{attr})$, let's consider two node objects src and dst, formally $src \in O$ and $dst \in O$ and exists e and e', such that

$$\gamma(e) = (src, t, n), \text{ where } t \in O \text{ and } n \in O$$
$$\gamma(e') = (n', t', dst), \text{ where } t' \in O \text{ and } n' \in O$$

Definition 3 (Path). *A path on the property graph G should be a sequence of objects*

$$src = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \dots \xrightarrow{e_k} v_k = dst$$

where $v_i, e_i \in O$ and for each $v_i \xrightarrow{e_i} v_{i+1}$, there exists $t_i \in O$ such that $\gamma(e_i) = (v_{i-1}, t_i, v_i)$.

For edge object e with a relation $\gamma(e) = (v, t, v')$, we define a function `TYPE` to get the type object of e.

$$\texttt{TYPE}(e) = t$$

We define a function λ on the path p to get a sequence of objects and labels. For path p:

$$src = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \dots \xrightarrow{e_k} v_k = dst$$

We have

$$\lambda(p) = \texttt{label}(v_0)\texttt{TYPE}(e_1)\texttt{label}(v_1)\texttt{TYPE}(e_2)\dots\texttt{TYPE}(e_k)\texttt{label}(v_k).$$

Semantics . Assume we have a property graph $G = (O, \gamma, \texttt{label}, \texttt{attr})$, a path p on G and a parameterized regular expression e, we define p satisfy the regular constraint of e as following:

- $\lambda(p) \in \texttt{Lang}(fst(e))$
- For each atomic ϕ in $snd(e)$, there exists an assignment A for global parameters and an object o, such that

$$A \models \phi_i[x/\texttt{attr}(o, x)]$$

References

[1] Vrgoč, D., Rojas, C., Angles, R., Arenas, M., Arroyuelo, D., Buil-Aranda, C., Hogan, A., Navarro, G., Riveros, C., and Romero, J. MillenniumDB: An Open-Source Graph Database System. *Data Intelligence 5*, 3 (08 2023), 560–610.