Assume that the database uses a binary EDB predicate *e* to store a directed graph. Try to express the following properties in semi-positive Datalog programs with a successor ordering, or explain why this is not possible.

1. The database contains an even number of elements.

$$\begin{aligned}
\mathit{Odd}(x) &\leftarrow \mathit{min}(x) \\
\mathit{Even}(y) &\leftarrow \mathit{Odd}(x) \wedge \mathit{succ}(x, y) \\
\mathit{Odd}(y) &\leftarrow \mathit{Even}(x) \wedge \mathit{succ}(x, y) \\
\mathit{Accept} &\leftarrow \mathit{Even}(x) \wedge \mathit{max}(x)
\end{aligned}$$

Assume that the database uses a binary EDB predicate *e* to store a directed graph. Try to express the following properties in semi-positive Datalog programs with a successor ordering, or explain why this is not possible.

2. The graph contains a node with two outgoing edges.

$$\mathit{Desc}(x, y) \leftarrow \mathit{succ}(x, y)$$

$$\mathit{Desc}(x, z) \leftarrow \mathit{Desc}(x, y) \wedge \mathit{succ}(y, z)$$

$$x \not\approx y \leftarrow \mathit{Desc}(x, y)$$

$$x \not\approx y \leftarrow \mathit{Desc}(y, x)$$

$$\mathit{Accept} \leftarrow e(x, y) \wedge e(x, z) \wedge y \not\approx z$$

Assume that the database uses a binary EDB predicate *e* to store a directed graph. Try to express the following properties in semi-positive Datalog programs with a successor ordering, or explain why this is not possible.

3. The graph is 3-colourable.

This is not expressible unless $\mathrm{P} = \mathrm{NP}$, since 3-colourability is NP-complete.

Theorem 13.3: A Boolean query mapping defines a language in P if and only if it can be described by a query in semipositive Datalog with a successor ordering.

4. The graph is *not* connected (*).

$$
\begin{aligned}
\mathit{Path}(x, y, \ell) &\leftarrow \mathit{e}(x, y) \wedge \mathit{min}(\ell) \\
\mathit{Path}(x, y, \ell) &\leftarrow \mathit{e}(y, x) \wedge \mathit{min}(\ell) \\
\mathit{Path}(x, z, \ell + 1) &\leftarrow \mathit{Path}(x, y, \ell) \wedge \mathit{e}(y, z) \wedge \mathit{succ}(\ell, \ell + 1) \\
\mathit{NoPathVia}(x, x, y, \ell) &\leftarrow \neg \mathit{edge}(x, y) \wedge \mathit{min}(\ell) \\
\mathit{NoPathVia}(x, y, z, \ell + 1) &\leftarrow \mathit{Path}(x, y, \ell) \wedge \neg \mathit{e}(y, z) \wedge \mathit{succ}(\ell, \ell + 1) \\
\mathit{NoPathVia}(x, z, y, \ell + 1) &\leftarrow \mathit{NoPath}(x, z, \ell) \\
\mathit{NoPathViaIt}(x, z, y, \ell) &\leftarrow \mathit{NoPathVia}(x, z, y, \ell) \wedge \mathit{min}(z) \\
\mathit{NoPathViaIt}(x, w, z, \ell) &\leftarrow \mathit{NoPathViaIt}(x, z, y, \ell) \wedge \mathit{succ}(z, w) \wedge \mathit{NoPathVia}(x, w, y, \ell) \\
\mathit{NoPath}(x, y, \ell) &\leftarrow \mathit{NoPathViaIt}(x, z, y, \ell) \wedge \mathit{max}(z) \\
\mathit{NoPathIt}(x, y, \ell) &\leftarrow \mathit{NoPath}(x, y, \ell) \wedge \mathit{min}(\ell) \\
\mathit{NoPathIt}(x, y, \ell + 1) &\leftarrow \mathit{NoPathIt}(x, y, \ell) \wedge \mathit{succ}(\ell, \ell + 1) \wedge \mathit{NoPath}(x, y, \ell + 1) \\
\mathit{Accept} &\leftarrow \mathit{NoPathIt}(x, y, \ell) \wedge \mathit{max}(\ell)
\end{aligned}
$$

5. The graph does not contain a node with two outgoing edges.

IDB predicates:

- *E0*(x, y): There are no edges from x into an element in $\{min, \ldots, y\}$.
- *E1*(x, y): There is exactly one outgoing edge from x into some node in $\{min, \ldots, y\}$.
- *R*(x): The set of all nodes with at most one outgoing edge.
- *S*(x): Used to iterate over all nodes; the iteration is successful if all nodes are in R.

$$\mathit{E0}(x,k) \leftarrow \mathit{min}(k) \wedge \neg e(x,k) \qquad \mathit{E0}(x,k+1) \leftarrow \mathit{E0}(x,k) \wedge \mathit{succ}(k,k+1) \wedge \neg e(x,k+1)$$

$$\mathit{E1}(x,k) \leftarrow \mathit{min}(k) \wedge e(x,k) \qquad \mathit{E1}(x,k+1) \leftarrow \mathit{E0}(x,k) \wedge \mathit{succ}(k,k+1) \wedge e(x,k+1)$$

$$\mathit{E1}(x,k+1) \leftarrow \mathit{E1}(x,k) \wedge \mathit{succ}(k,k+1) \wedge \neg e(x,k+1)$$

$$R(x) \leftarrow \mathit{E0}(x,k) \wedge \mathit{max}(k) \qquad S(x) \leftarrow \mathit{min}(x) \wedge R(x)$$

$$R(x) \leftarrow \mathit{E1}(x,k) \wedge \mathit{max}(k) \qquad S(x) \leftarrow \mathit{succ}(y,x) \wedge S(y) \wedge R(x)$$

$$\mathit{Accept} \leftarrow S(x) \wedge \mathit{max}(x)$$

6. The graph is a chain.

$$\mathit{Accept} \leftarrow \mathit{AtMostOneEdge} \land \mathit{ConnectedGraph} \land \mathit{NoCycle}$$

$$\mathit{Con}(x, x) \leftarrow$$

$$\mathit{Con}(x, y) \leftarrow \mathit{Con}(y, x)$$

$$\mathit{Con}(x, y) \leftarrow \mathit{Con}(x, z) \land \mathit{Con}(z, y)$$

$$\mathit{Con}(y, x) \leftarrow e(x, y)$$

$$\mathit{Con}(x, y) \leftarrow e(x, y)$$

$$\mathit{ConToPred}(y) \leftarrow \mathit{min}(x) \land \mathit{succ}(x, y) \land \mathit{Con}(x, y)$$

$$\mathit{ConToPred}(y) \leftarrow \mathit{ConToPred}(x) \land \mathit{succ}(x, y) \land \mathit{Con}(x, y)$$

$$\mathit{ConToPred}(y) \leftarrow \mathit{ConToPred}(x) \land \mathit{max}(x) \land \mathit{min}(y) \land \mathit{Con}(x, y)$$

$$\mathit{ConnectedGraph} \leftarrow \mathit{ConToPred}(x) \land \mathit{min}(x)$$

6. The graph is a chain.

$$\textit{Accept} \leftarrow \textit{AtMostOneEdge} \wedge \textit{ConnectedGraph} \wedge \textit{NoCycle}$$

- *NoInEdge*(k, x): There are no edges from an element in $\{min, \dots, k\}$ into x.
- *NoOutEdge*(x, k): There are no edges from x into an element in $\{min, \dots, k\}$.

$$\begin{aligned}
\textit{NoInEdg}(k, x) &\leftarrow \textit{min}(k) \wedge \neg \textit{e}(k, x) \\
\textit{NoInEdg}(k + 1, x) &\leftarrow \textit{NoInEdg}(k, x) \wedge \textit{succ}(k, k + 1) \wedge \neg \textit{e}(k + 1, x) \\
\exists \textit{NodeWithNoInEdge} &\leftarrow \textit{max}(k) \wedge \textit{NoInEdg}(k, x) \\
\textit{NoOutEdg}(x, k) &\leftarrow \textit{min}(k) \wedge \neg \textit{e}(x, k) \\
\textit{NoOutEdg}(x, k) &\leftarrow \textit{succ}(k + 1, k) \wedge \textit{NoOutEdg}(x, k + 1) \wedge \neg \textit{e}(x, k) \\
\exists \textit{NodeWithNoOutEdge} &\leftarrow \textit{max}(k) \wedge \textit{NoOutEdg}(x, k) \\
\textit{NoCycle} &\leftarrow \exists \textit{NodeWithNoInEdge} \wedge \exists \textit{NodeWithNoOutEdge}
\end{aligned}$$