**Exercise 1**

**Exercise.** Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.
Exercise 1

Exercise. Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

Definition (Lecture 12, Slide 17)

- A Datalog rule $H \leftarrow B$ is safe if all variables in $H$ also occur in $B$.
- A Datalog program $P$ is safe if all rules $r \in P$ are safe.
Exercise 1

Exercise. Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

Definition (Lecture 12, Slide 17)

- A Datalog rule $H \leftarrow B$ is safe if all variables in $H$ also occur in $B$.
- A Datalog program $P$ is safe if all rules $r \in P$ are safe.

Solution.
**Exercise 1**

**Exercise.** Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

**Definition (Lecture 12, Slide 17)**

- A Datalog rule $H \leftarrow B$ is **safe** if all variables in $H$ also occur in $B$.
- A Datalog program $P$ is **safe** if all rules $r \in P$ are safe.

**Solution.**

- Consider a (possibly unsafe) Datalog program $P$. 
Exercise 1

**Exercise.** Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

**Definition (Lecture 12, Slide 17)**

- A Datalog rule $H \leftarrow B$ is **safe** if all variables in $H$ also occur in $B$.
- A Datalog program $P$ is **safe** if all rules $r \in P$ are safe.

**Solution.**

- Consider a (possibly unsafe) Datalog program $P$.
- We define a new Datalog program $P'$:
Exercise 1

**Exercise.** Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

**Definition (Lecture 12, Slide 17)**

- A Datalog rule \( H \leftarrow B \) is **safe** if all variables in \( H \) also occur in \( B \).
- A Datalog program \( P \) is **safe** if all rules \( r \in P \) are safe.

**Solution.**

- Consider a (possibly unsafe) Datalog program \( P \).
- We define a new Datalog program \( P' \):
  - we add a fresh predicate Top,
Exercise 1

**Exercise.** Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

**Definition (Lecture 12, Slide 17)**

- A Datalog rule $H \leftarrow B$ is **safe** if all variables in $H$ also occur in $B$.
- A Datalog program $P$ is **safe** if all rules $r \in P$ are safe.

**Solution.**

- Consider a (possibly unsafe) Datalog program $P$.
- We define a new Datalog program $P'$:
  - we add a fresh predicate Top,
  - for every $\ell$-ary EDB predicate $r$ occurring in $P$ and all $1 \leq i \leq \ell$, we add a new rule $\text{Top}(x_i) \leftarrow r(x_1, \ldots, x_\ell)$,
Exercise 1

**Exercise.** Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

**Definition (Lecture 12, Slide 17)**

- A Datalog rule $H \leftarrow B$ is **safe** if all variables in $H$ also occur in $B$.
- A Datalog program $P$ is **safe** if all rules $r \in P$ are safe.

**Solution.**

- Consider a (possibly unsafe) Datalog program $P$.
- We define a new Datalog program $P'$:
  - we add a fresh predicate Top,
  - for every $\ell$-ary EDB predicate $r$ occurring in $P$ and all $1 \leq i \leq \ell$, we add a new rule $\text{Top}(x_i) \leftarrow r(x_1, \ldots, x_\ell)$,
  - for every constant $c$ occurring in $P$, we add a new fact $\text{Top}(c)$,
Exercise 1

**Exercise.** Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

**Definition (Lecture 12, Slide 17)**

- A Datalog rule $H \leftarrow B$ is **safe** if all variables in $H$ also occur in $B$.
- A Datalog program $P$ is **safe** if all rules $r \in P$ are safe.

**Solution.**

- Consider a (possibly unsafe) Datalog program $P$.
- We define a new Datalog program $P'$:
  - we add a fresh predicate Top,
  - for every $\ell$-ary EDB predicate $r$ occurring in $P$ and all $1 \leq i \leq \ell$, we add a new rule $\text{Top}(x_i) \leftarrow r(x_1, \ldots, x_i)$,
  - for every constant $c$ occurring in $P$, we add a new fact $\text{Top}(c)$,
  - for every rule $(H \leftarrow B)[x_1, \ldots, x_n] \in P$, we add the rule $H \leftarrow B \land \text{Top}(x_1) \land \cdots \land \text{Top}(x_n)$.

Then for every fact $\phi$ over the signature of $P$, we have that $P'$ entails $\phi$ over an instance $D$ if $P$ entails $\phi$ over $D$.

The size of $P'$ is polynomial in the size of $P$, and $P'$ is safe.
Exercise 1

Exercise. Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

Definition (Lecture 12, Slide 17)

▶ A Datalog rule $H \leftarrow B$ is safe if all variables in $H$ also occur in $B$.
▶ A Datalog program $P$ is safe if all rules $r \in P$ are safe.

Solution.

▶ Consider a (possibly unsafe) Datalog program $P$.
▶ We define a new Datalog program $P'$:
  ▶ we add a fresh predicate Top,
  ▶ for every $\ell$-ary EDB predicate $r$ occurring in $P$ and all $1 \leq i \leq \ell$, we add a new rule $\text{Top}(x_i) \leftarrow r(x_1, \ldots, x_\ell)$,
  ▶ for every constant $c$ occurring in $P$, we add a new fact $\text{Top}(c)$,
  ▶ for every rule $(H \leftarrow B)[x_1, \ldots, x_n] \in P$, we add the rule $H \leftarrow B \land \text{Top}(x_1) \land \cdots \land \text{Top}(x_\ell)$.
▶ Then for every fact $\varphi$ over the signature of $P$, we have that $P'$ entails $\varphi$ over an instance $D$ iff $P$ entails $\varphi$ over $D$. 
Exercise 1

**Exercise.** Show that any Datalog program can be expressed as a safe Datalog program that is polynomial in the size of the original program and given schema.

**Definition (Lecture 12, Slide 17)**

- A Datalog rule $H \leftarrow B$ is **safe** if all variables in $H$ also occur in $B$.
- A Datalog program $P$ is **safe** if all rules $r \in P$ are safe.

**Solution.**

- Consider a (possibly unsafe) Datalog program $P$.
- We define a new Datalog program $P'$:
  - we add a fresh predicate Top,
  - for every $\ell$-ary EDB predicate $r$ occurring in $P$ and all $1 \leq i \leq \ell$, we add a new rule $\text{Top}(x_i) \leftarrow r(x_1, \ldots, x_\ell)$,
  - for every constant $c$ occurring in $P$, we add a new fact $\text{Top}(c)$,
  - for every rule $(H \leftarrow B)[x_1, \ldots, x_n] \in P$, we add the rule $H \leftarrow B \land \text{Top}(x_1) \land \cdots \land \text{Top}(x_\ell)$.

- Then for every fact $\varphi$ over the signature of $P$, we have that $P'$ entails $\varphi$ over an instance $D$ iff $P$ entails $\varphi$ over $D$.
- The size of $P'$ is polynomial in the size of $P$, and $P'$ is safe.
Exercise 2

**Exercise.** Assume that the database uses a binary EDB predicate `edge` 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.
2. The graph contains a node with two outgoing edges.
3. The graph is 3-colourable.
4. The graph is *not* connected (*). 
5. The graph does not contain a node with two outgoing edges.
6. The graph is a chain.
Exercise 2

**Exercise.** Assume that the database uses a binary EDB predicate `edge` 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.
2. The graph contains a node with two outgoing edges.
3. The graph is 3-colourable.
4. The graph is *not* connected (*).
5. The graph does not contain a node with two outgoing edges.
6. The graph is a chain.

**Solution.**
Exercise 2

Exercise. Assume that the database uses a binary EDB predicate edge 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.
2. The graph contains a node with two outgoing edges.
3. The graph is 3-colourable.
4. The graph is not connected (*).
5. The graph does not contain a node with two outgoing edges.
6. The graph is a chain.

Solution.

1.

\[
\begin{align*}
\text{Odd}(x) & \leftarrow \text{first}(x) \\
\text{Odd}(y) & \leftarrow \text{Even}(x), \text{succ}(x, y) \\
\text{Even}(y) & \leftarrow \text{Odd}(x), \text{succ}(x, y) \\
\text{EvenParity()} & \leftarrow \text{Even}(x), \text{last}(x)
\end{align*}
\]
Exercise 2

**Exercise.** Assume that the database uses a binary EDB predicate `edge` 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.
2. The graph contains a node with two outgoing edges.
3. The graph is 3-colourable.
4. The graph is *not* connected (*).
5. The graph does not contain a node with two outgoing edges.
6. The graph is a chain.

**Solution.**

2.

\[
\begin{align*}
<(x, y) & \leftarrow \text{succ}(x, y) \\
<(x, z) & \leftarrow <(x, y), \text{succ}(y, z) \\
<>(y, z) & \leftarrow <>((y, x), <(x, y)) \\
\text{TwoOutgoingEdges()} & \leftarrow \text{edge}(x, y), \text{edge}(x, z), <>((y, z))
\end{align*}
\]
Exercise 2

Exercise. Assume that the database uses a binary EDB predicate $\text{edge}$ 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.
2. The graph contains a node with two outgoing edges.
3. The graph is 3-colourable.
4. The graph is not connected (*).
5. The graph does not contain a node with two outgoing edges.
6. The graph is a chain.

Solution.

2.

\[
\begin{align*}
<(x, y) & \leftarrow \text{succ}(x, y) \\
<(x, z) & \leftarrow <(x, y), \text{succ}(y, z) \\
<>(x, y), <>(y, x) & \leftarrow <(x, y) \\
\text{TwoOutgoingEdges}(\cdot) & \leftarrow \text{edge}(x, y), \text{edge}(x, z), <> (y, z)
\end{align*}
\]

3. This is (most likely) not expressible (unless $P = NP$), since 3-colourability is $NP$-complete and Datalog has $P$ data complexity.
Exercise 2

**Exercise.** Assume that the database uses a binary EDB predicate `edge` 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.
2. The graph contains a node with two outgoing edges.
3. The graph is 3-colourable.
4. The graph is not connected (*).
5. The graph does not contain a node with two outgoing edges.
6. The graph is a chain.

**Solution.**

4.  
   - C(x, y) \( x \) and \( y \) are in the same connected component
   - D(x, y, k) \( x \) and \( y \) are not reachable via a path of length at most \( k \)
   - N(x, y, z, k) there is no path of length \( k + 1 \) from \( x \) to \( z \) via \( y \)

\[
\begin{align*}
\text{C}(x,x) & \leftarrow \\
\text{C}(x,z) & \leftarrow \text{C}(x,y), \text{C}(y,z) \\
\text{N}(x,y,z,k) & \leftarrow \text{first}(y), \text{D}(x,y,k) \\
\text{N}(x,y',z,k) & \leftarrow \text{succ}(y,y'), \text{N}(x,y,z,k), \text{D}(x,y',k) \\
\text{D}(x,z,k') & \leftarrow \text{succ}(k,k'), \text{D}(x,z,k), \text{last}(y), \text{N}(x,y,z,k) \\
\text{C}(x,y), \text{C}(y,x) & \leftarrow \text{edge}(x,y) \\
\text{D}(x,y,\ell), \text{D}(y,x,\ell) & \leftarrow \neg \text{edge}(x,y), \text{first}(\ell), \text{<>}(x,y) \\
\text{N}(x,y,z,k) & \leftarrow \text{first}(y), \text{D}(y,z,k) \\
\text{N}(x,y',z,k) & \leftarrow \text{succ}(y,y'), \text{N}(x,y,z,k), \text{D}(y',z,k) \\
\text{Ans}() & \leftarrow \text{D}(x,y,k), \text{last}(k)
\end{align*}
\]
Exercise 2

**Exercise.** Assume that the database uses a binary EDB predicate \( \text{edge} \) 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.
2. The graph contains a node with two outgoing edges.
3. The graph is 3-colourable.
4. The graph is *not* connected (*).
5. The graph does not contain a node with two outgoing edges.
6. The graph is a chain.

**Solution.**

5.

\[
\begin{align*}
\text{oneEdge}(x, y) & \leftarrow \text{first}(y), \text{edge}(x, y) & \text{noEdge}(x, y) & \leftarrow \text{first}(y), \neg \text{edge}(x, y) \\
\text{oneEdge}(x, z) & \leftarrow \text{noEdge}(x, y), \text{succ}(y, z), \text{edge}(x, z) & \text{noEdge}(x, z) & \leftarrow \text{noEdge}(x, y), \text{succ}(y, z), \neg \text{edge}(x, z) \\
\text{oneEdge}(x, z) & \leftarrow \text{oneEdge}(x, y), \text{succ}(y, z), \neg \text{edge}(x, z) & & \\
\text{r}(x) & \leftarrow \text{last}(y), \text{noEdge}(x, y) & \text{s}(x) & \leftarrow \text{first}(x), \text{r}(x) \\
\text{r}(x) & \leftarrow \text{last}(y), \text{oneEdge}(x, y) & \text{s}(y) & \leftarrow \text{succ}(x, y), \text{s}(x), \text{r}(y) \\
\text{NoTwoOutEdges}() & \leftarrow \text{s}(x), \text{last}(x)
\end{align*}
\]
Exercise 2

**Exercise.** Assume that the database uses a binary EDB predicate *edge* 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.
2. The graph contains a node with two outgoing edges.
3. The graph is 3-colourable.
4. The graph is *not* connected (*). 
5. The graph does not contain a node with two outgoing edges.
6. The graph is a chain.

**Solution.**

6.

\[
\text{Chain}(()) \leftarrow \text{Connected}(), \text{NoTwoInEdges}(), \text{NoTwoOutEdges}(), \text{NoCycle}()
\]

\[
\begin{align*}
C(x), R(x) & \leftarrow \text{first}(x) &
C(y) & \leftarrow C(x), \text{edge}(x, y) \\
R(x) & \leftarrow R(x), \text{succ}(x, y), C(y) &
C(y) & \leftarrow C(x), \text{edge}(y, x) \\
\end{align*}
\]

\[
\begin{align*}
\text{Connected}() & \leftarrow \text{last}(x), R(x) \\
\text{nI}(x, y) & \leftarrow \text{first}(x), \neg \text{edge}(x, y) &
\text{nO}(x, y) & \leftarrow \text{first}(x), \neg \text{edge}(y, x) \\
\text{nI}(x', y) & \leftarrow \text{succ}(x, x'), \text{nI}(x, y), \neg \text{edge}(x', y) &
\text{nO}(x', y) & \leftarrow \text{succ}(x, x'), \text{nO}(x, y), \neg \text{edge}(y, x') \\
\text{NoCycle}() & \leftarrow \text{last}(x), \text{nI}(x, y), \text{nO}(x, z)
\end{align*}
\]
Exercise 3

**Exercise.** A Horn logic program is in $\text{propHorn2}$ if every rule it contains is of the form $H \leftarrow H$ or $H \leftarrow B_1 \land B_2$.

It was claimed that entailment checking in $\text{propHorn2}$ is $\text{P}$-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in $\text{propHorn2}$. Argue how this reduction can be accomplished in logarithmic space.

Solution. Let $P$ be a propositional Horn logic program. Then, let $P'$ be the propositional Horn logic program such that, for all formulas $\text{Body} \rightarrow H \in P$,

- if $\text{Body} = B$ consists of a single atom, then add $B \land B \rightarrow H \in P'$,
- if $\text{Body} = B_1 \land \ldots \land B_n$ with $n \geq 3$, then add $B_1 \land B_2 \rightarrow F_2, F_2 \land B_3 \rightarrow F_3, \ldots, F_{n-1} \land B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
- otherwise, add $\text{Body} \rightarrow H \in P'$.

For all propositional variables $V$ occurring in $P$, we have that $P | V = \bot$ if and only if $P | V' = \bot$.

The program $P'$ can be computed with a $\text{LogSpace}$ transducer:

- count the number of body atoms to generate these rules,
- count the number of rules to have fresh identifiers for every newly translated rule, and
- count the number of any propositional variable name to have unique identifiers.
Exercise 3

**Exercise.** A Horn logic program is in propHorn2 if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \land B_2$. It was claimed that entailment checking in propHorn2 is $P$-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in propHorn2. Argue how this reduction can be accomplished in logarithmic space.

**Solution.**
Exercise 3

Exercise. A Horn logic program is in propHorn2 if every rule it contains is of the form \( H \leftarrow \) or \( H \leftarrow B_1 \land B_2 \).

It was claimed that entailment checking in propHorn2 is \( P \)-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in propHorn2. Argue how this reduction can be accomplished in logarithmic space.

Solution.

▶ Let \( P \) be a propositional Horn logic program.
Exercise 3

**Exercise.** A Horn logic program is in propHorn2 if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \land B_2$. It was claimed that entailment checking in propHorn2 is $P$-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in propHorn2. Argue how this reduction can be accomplished in logarithmic space.

**Solution.**

- Let $P$ be a propositional Horn logic program.
- Then, let $P'$ be the proposition Horn logic program such that, for all formulas $\text{Body} \rightarrow H \in P$, 

  - If $\text{Body} = B$ consists of a single atom, then add $B \land B \rightarrow H \in P'$.
  - If $\text{Body} = B_1 \land \ldots \land B_n$ with $n \geq 3$, then add $B_1 \land B_2 \rightarrow F_2$, $F_2 \land B_3 \rightarrow F_3$, ..., $F_{n-1} \land B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - Otherwise, add $\text{Body} \rightarrow H \in P'$.
Exercise 3

**Exercise.** A Horn logic program is in *propHorn2* if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \land B_2$. It was claimed that entailment checking in *propHorn2* is P-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in *propHorn2*. Argue how this reduction can be accomplished in logarithmic space.

**Solution.**

- Let $P$ be a propositional Horn logic program.
- Then, let $P'$ be the proposition Horn logic program such that, for all formulas $\text{Body} \rightarrow H \in P$,
  - if $\text{Body} = B$ consists of a single atom, then add $B \land B \rightarrow H \in P'$.
Exercise 3

Exercise. A Horn logic program is in propHorn2 if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \land B_2$. It was claimed that entailment checking in propHorn2 is $P$-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in propHorn2. Argue how this reduction can be accomplished in logarithmic space.

Solution.

- Let $P$ be a propositional Horn logic program.
- Then, let $P'$ be the proposition Horn logic program such that, for all formulas $\text{Body} \rightarrow H \in P$,
  - if $\text{Body} = B$ consists of a single atom, then add $B \land B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \land \ldots \land B_n$ with $n \geq 3$, then add $B_1 \land B_2 \rightarrow F_2, F_2 \land B_3 \rightarrow F_3, \ldots, F_{n-1} \land B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
Exercise 3

**Exercise.** A Horn logic program is in propHorn2 if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \land B_2$. It was claimed that entailment checking in propHorn2 is $P$-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in propHorn2. Argue how this reduction can be accomplished in logarithmic space.

**Solution.**

- Let $P$ be a propositional Horn logic program.
- Then, let $P'$ be the propositional Horn logic program such that, for all formulas $\text{Body} \rightarrow H \in P$,
  - if $\text{Body} = B$ consists of a single atom, then add $B \land B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \land \ldots \land B_n$ with $n \geq 3$, then add $B_1 \land B_2 \rightarrow F_2, F_2 \land B_3 \rightarrow F_3, \ldots, F_{n-1} \land B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - otherwise, add $\text{Body} \rightarrow H \in P'$.
Exercise. A Horn logic program is in $\text{propHorn2}$ if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \land B_2$.

It was claimed that entailment checking in $\text{propHorn2}$ is P-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in $\text{propHorn2}$. Argue how this reduction can be accomplished in logarithmic space.

Solution.

- Let $P$ be a propositional Horn logic program.
- Then, let $P'$ be the proposition Horn logic program such that, for all formulas $\text{Body} \rightarrow H \in P$,
  - if $\text{Body} = B$ consists of a single atom, then add $B \land B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \land \ldots \land B_n$ with $n \geq 3$, then add $B_1 \land B_2 \rightarrow F_2, F_2 \land B_3 \rightarrow F_3, \ldots, F_{n-1} \land B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - otherwise, add $\text{Body} \rightarrow H \in P'$.
- For all propositional variables $V$ occurring in $P$, we have that $P \models V$ if and only if $P \models V'$. 
Exercise 3

**Exercise.** A Horn logic program is in \textit{propHorn2} if every rule it contains is of the form \( H \leftarrow \) or \( H \leftarrow B_1 \land B_2 \).

It was claimed that entailment checking in \textit{propHorn2} is \textsc{P}-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in \textit{propHorn2}. Argue how this reduction can be accomplished in logarithmic space.

**Solution.**

- Let \( P \) be a propositional Horn logic program.
- Then, let \( P' \) be the proposition Horn logic program such that, for all formulas \( \text{Body} \rightarrow H \in P \),
  - if \( \text{Body} = B \) consists of a single atom, then add \( B \land B \rightarrow H \in P' \),
  - if \( \text{Body} = B_1 \land \ldots \land B_n \) with \( n \geq 3 \), then add \( B_1 \land B_2 \rightarrow F_2, F_2 \land B_3 \rightarrow F_3, \ldots, F_{n-1} \land B_n \rightarrow H \in P' \) where \( F_2, \ldots, F_{n-1} \) are fresh propositional variables.
  - otherwise, add \( \text{Body} \rightarrow H \in P' \).
- For all propositional variables \( V \) occurring in \( P \), we have that \( P \models V \) if and only if \( P \models V' \).
- The program \( P' \) can be computed with a \textsc{LogSpace} transducer:
Exercise 3

**Exercise.** A Horn logic program is in *propHorn2* if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \land B_2$.

It was claimed that entailment checking in *propHorn2* is $P$-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in *propHorn2*. Argue how this reduction can be accomplished in logarithmic space.

**Solution.**

- Let $P$ be a propositional Horn logic program.
- Then, let $P'$ be the proposition Horn logic program such that, for all formulas $\text{Body} \rightarrow H \in P$,
  - if $\text{Body} = B$ consists of a single atom, then add $B \land B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \land \ldots \land B_n$ with $n \geq 3$, then add $B_1 \land B_2 \rightarrow F_2, F_2 \land B_3 \rightarrow F_3, \ldots, F_{n-1} \land B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - otherwise, add $\text{Body} \rightarrow H \in P'$.
- For all propositional variables $V$ occurring in $P$, we have that $P \models V$ if and only if $P' \models V'$.
- The program $P'$ can be computed with a $\text{LOGSPACE}$ transducer:
  - count number of body atoms to generate these rules
Exercise. A Horn logic program is in \textit{propHorn2} if every rule it contains is of the form \( H \leftarrow \) or \( H \leftarrow B_1 \land B_2 \).

It was claimed that entailment checking in \textit{propHorn2} is \( P \)-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in \textit{propHorn2}. Argue how this reduction can be accomplished in logarithmic space.

Solution.

- Let \( P \) be a propositional Horn logic program.
- Then, let \( P' \) be the proposition Horn logic program such that, for all formulas \( \text{Body} \rightarrow H \in P \),
  - if \( \text{Body} = B \) consists of a single atom, then add \( B \land B \rightarrow H \in P' \),
  - if \( \text{Body} = B_1 \land \ldots \land B_n \) with \( n \geq 3 \), then add \( B_1 \land B_2 \rightarrow F_2, F_2 \land B_3 \rightarrow F_3, \ldots, F_{n-1} \land B_n \rightarrow H \in P' \) where \( F_2, \ldots, F_{n-1} \) are fresh propositional variables.
  - otherwise, add \( \text{Body} \rightarrow H \in P' \).
- For all propositional variables \( V \) occurring in \( P \), we have that \( P \models V \) if and only if \( P' \models V' \).
- The program \( P' \) can be computed with a \texttt{LogSpace} transducer:
  - count number of body atoms to generate these rules
  - count number of rules to have fresh identifiers for every newly translated rule, and
Exercise 3

**Exercise.** A Horn logic program is in $\text{propHorn2}$ if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \land B_2$.

It was claimed that entailment checking in $\text{propHorn2}$ is $P$-hard. To support this claim, explain how entailment in propositional Horn logic can be reduced to entailment in $\text{propHorn2}$. Argue how this reduction can be accomplished in logarithmic space.

**Solution.**

- Let $P$ be a propositional Horn logic program.
- Then, let $P'$ be the proposition Horn logic program such that, for all formulas $\text{Body} \rightarrow H \in P$,
  - if $\text{Body} = B$ consists of a single atom, then add $B \land B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \land \ldots \land B_n$ with $n \geq 3$, then add $B_1 \land B_2 \rightarrow F_2$, $F_2 \land B_3 \rightarrow F_3$, ..., $F_{n-1} \land B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables,
  - otherwise, add $\text{Body} \rightarrow H \in P'$.
- For all propositional variables $V$ occurring in $P$, we have that $P \models V$ if and only if $P' \models V'$.
- The program $P'$ can be computed with a $\text{LOGSPACE}$ transducer:
  - count number of body atoms to generate these rules
  - count number of rules to have fresh identifiers for every newly translated rule, and
  - count the of any propositional variable name to have unique identifiers
Exercise 4

**Exercise.** Prove that entailment checking in propositional Horn logic is $P$-hard.
Exercise 4

Exercise. Prove that entailment checking in propositional Horn logic is P-hard.

Solution.
Exercise 4

Exercise. Prove that entailment checking in propositional Horn logic is \( P \)-hard.

Solution.

Consider a \( P \)-TM \( M = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle \) and an input word \( w = w_1, \ldots, w_n \in \Sigma^* \).
Exercise 4

**Exercise.** Prove that entailment checking in propositional Horn logic is $P$-hard.

**Solution.**

- Consider a $P$-TM $M = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle$ and an input word $w = w_1, \ldots, w_n \in \Sigma^*$.
- We define a Datalog program $P$ such that $P$ entails a predicate $\text{Accept()}$ iff the TM $M$ accepts $w$.
Exercise 4

**Exercise.** Prove that entailment checking in propositional Horn logic is $\mathbf{P}$-hard.

**Solution.**

- Consider a $\mathbf{P}$-TM $M = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle$ and an input word $w = w_1, \ldots, w_n \in \Sigma^*$.
- We define a Datalog program $P$ such that $P$ entails a predicate $\text{Accept}()$ iff the TM $M$ accepts $w$.
- Since $M$ is polynomial, $M$ halts after at most $n^k$ steps for some $k \geq 0$. 
Exercise 4

**Exercise.** Prove that entailment checking in propositional Horn logic is \( P \)-hard.

**Solution.**

- Consider a \( P \)-TM \( M = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle \) and an input word \( w = w_1, \ldots, w_n \in \Sigma^* \).
- We define a Datalog program \( P \) such that \( P \) entails a predicate \( \text{Accept}() \) iff the TM \( M \) accepts \( w \).
- Since \( M \) is polynomial, \( M \) halts after at most \( n^k \) steps for some \( k \geq 0 \).
- Constants:
  - \( \text{cell}_{i,j} \) for all \( 1 \leq i \leq j \leq n^k + 1 \), and
  - all elements \( q \in Q \) and \( \gamma \in \Gamma \)
**Exercise.** Prove that entailment checking in propositional Horn logic is $P$-hard.

**Solution.**

- Consider a $P$-TM $\mathcal{M} = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle$ and an input word $w = w_1, \ldots, w_n \in \Sigma^*$.
- We define a Datalog program $P$ such that $P$ entails a predicate $\text{Accept}()$ iff the TM $\mathcal{M}$ accepts $w$.
- Since $\mathcal{M}$ is polynomial, $\mathcal{M}$ halts after at most $n^k$ steps for some $k \geq 0$.
- **Constants:**
  - $\text{cell}_{i,j}$ for all $1 \leq i \leq j \leq n^k + 1$, and
  - all elements $q \in Q$ and $\gamma \in \Gamma$
- **Facts:**
  - $\text{right}(\text{cell}_{i,j}, \text{cell}_{i,j+1})$, $\text{future}(\text{cell}_{i,j}, \text{cell}_{i+1,j})$, for $1 \leq i \leq j \leq n^k$,
  - $S(\text{cell}_{0,i}, w_i)$ for $1 \leq i \leq n$, and $S(\text{cell}_{0,i}, b)$ for $n + 1 \leq i \leq n^k$, and
  - $T(\text{cell}_{0,0}, q_0)$
Exercise 4

**Exercise.** Prove that entailment checking in propositional Horn logic is \(P\)-hard.

**Solution.**

- Consider a \(P\)-TM \(M = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle\) and an input word \(w = w_1, \ldots, w_n \in \Sigma^*\).
- We define a Datalog program \(P\) such that \(P\) entails a predicate \(\text{Accept}()\) iff the TM \(M\) accepts \(w\).
- Since \(M\) is polynomial, \(M\) halts after at most \(n^k\) steps for some \(k \geq 0\).
- Constants:
  - \(\text{cell}_{i,j}\) for all \(1 \leq i \leq j \leq n^k + 1\), and
  - all elements \(q \in Q\) and \(\gamma \in \Gamma\)
- Facts:
  - \(\text{right}(\text{cell}_{i,j}, \text{cell}_{i,j+1}), \text{future}(\text{cell}_{i,j}, \text{cell}_{i+1,j})\), for \(1 \leq i \leq j \leq n^k\),
  - \(S(\text{cell}_{0,i}, w_i)\) for \(1 \leq i \leq n\), and \(S(\text{cell}_{0,i}, b)\) for \(n + 1 \leq i \leq n^k\), and
  - \(T(\text{cell}_{0,0}, q_0)\)
- Rules:
  - \(\text{Accept}() \leftarrow T(x, q_f)\).
  - \(\text{NTR}(z) \leftarrow T(x, y) \land \text{right}(x, z)\), \(\text{NTR}(y) \leftarrow \text{NTR}(x) \land \text{right}(x, y)\), \(\text{NTL}(z) \leftarrow T(x, y) \land \text{right}(z, x)\),
    \(\text{NTL}(x) \leftarrow \text{right}(x, y) \land \text{NTL}(y)\), \(\text{NT}(x) \leftarrow \text{NTR}(x)\), \(\text{NT}(x) \leftarrow \text{NTL}(x)\), \(S(y, z) \leftarrow \text{NT}(x) \land \text{future}(x, y) \land S(x, z)\),
  - \(T(z, q') \leftarrow T(x, q) \land S(x, \gamma) \land \text{future}(x, y) \land \text{right}(z, y)\), \(S(y, \gamma') \leftarrow T(x, q) \land S(x, \gamma) \land \text{future}(x, y)\) for all \(\langle q, \gamma, q', \gamma', L \rangle \in \delta\),
  - and similarly for all \(\langle q, \gamma, q', \gamma', R \rangle \in \delta\)
Exercise 4

Exercise. Prove that entailment checking in propositional Horn logic is $P$-hard.

Solution.

- Consider a $P$-TM $M = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle$ and an input word $w = w_1, \ldots , w_n \in \Sigma^*$.
- We define a Datalog program $P$ such that $P$ entails a predicate $Accept()$ iff the TM $M$ accepts $w$.
- Since $M$ is polynomial, $M$ halts after at most $n^k$ steps for some $k \geq 0$.

Constants:
- $cell_{i,j}$ for all $1 \leq i \leq j \leq n^k + 1$, and
- all elements $q \in Q$ and $\gamma \in \Gamma$

Facts:
- $right(cell_{i,j}, cell_{i,j+1})$, $future(cell_{i,j}, cell_{i+1,j})$, for $1 \leq i \leq j \leq n^k$,  
- $S(cell_{0,i}, w_i)$ for $1 \leq i \leq n$, and $S(cell_{0,i}, b)$ for $n + 1 \leq i \leq n^k$, and
- $T(cell_{0,0}, q_0)$

Rules:
- $Accept() \leftarrow T(x, q_f)$,
- $NTR(z) \leftarrow T(x, y) \land right(x, z)$, $NTR(y) \leftarrow NTR(x) \land right(x, y)$, $NTL(z) \leftarrow T(x, y) \land right(z, x)$,  
  $NTL(x) \leftarrow right(x, y) \land NTL(y)$, $NT(x) \leftarrow NTR(x)$, $NTL(x) \leftarrow NTL(x)$, $S(y, z) \leftarrow NT(x) \land future(x, y) \land S(x, z)$,  
- $T(z, q') \leftarrow T(x, q) \land S(x, y) \land future(x, y) \land right(z, y)$, $S(y, y') \leftarrow T(x, q) \land S(x, y) \land future(x, y)$ for all $\langle q, \gamma, q', \gamma', L \rangle \in \delta$,  
  and similarly for all $\langle q, \gamma, q', \gamma', R \rangle \in \delta$

- The grounding of $P$ is a Propositional Horn Logic program that is polynomial in the size of $P$ (which is polynomial in the size of $n$).
Exercise 4

**Exercise.** Prove that entailment checking in propositional Horn logic is \( \mathcal{P} \)-hard.

**Solution.**

- Consider a \( \mathcal{P} \)-TM \( M = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle \) and an input word \( w = w_1, \ldots, w_n \in \Sigma^* \).
- We define a Datalog program \( P \) such that \( P \) entails a predicate \( \text{Accept}() \) iff the TM \( M \) accepts \( w \).
- Since \( M \) is polynomial, \( M \) halts after at most \( n^k \) steps for some \( k \geq 0 \).

**Constants:**
- \( \text{cell}_{i,j} \) for all \( 1 \leq i \leq j \leq n^k + 1 \), and
- all elements \( q \in Q \) and \( \gamma \in \Gamma \)

**Facts:**
- \( \text{right}(\text{cell}_{i,j}, \text{cell}_{i,j+1}), \text{future}(\text{cell}_{i,j}, \text{cell}_{i+1,j}) \), for \( 1 \leq i \leq j \leq n^k \),
- \( S(\text{cell}_{0,i}, w_i) \) for \( 1 \leq i \leq n \), and \( S(\text{cell}_{0,i}, b) \) for \( n + 1 \leq i \leq n^k \), and
- \( T(\text{cell}_{0,0}, q_0) \)

**Rules:**
- \( \text{Accept}() \leftarrow T(x, q_f) \),
- \( \text{NTR}(z) \leftarrow T(x, y) \land \text{right}(x, z), \text{NTR}(y) \leftarrow \text{NTR}(x) \land \text{right}(x, y), \text{NTL}(z) \leftarrow T(x, y) \land \text{right}(z, x) \),
- \( \text{NTL}(x) \leftarrow \text{right}(x, y) \land \text{NTL}(y), \text{NT}(x) \leftarrow \text{NTR}(x), \text{NT}(x) \leftarrow \text{NTL}(x), \text{S}(y, z) \leftarrow \text{NT}(x) \land \text{future}(x, y) \land S(x, z) \),
- \( T(z, q') \leftarrow T(x, q) \land S(x, \gamma) \land \text{future}(x, y) \land \text{right}(z, y), S(y, \gamma') \leftarrow T(x, q) \land S(x, \gamma) \land \text{future}(x, y) \) for all \( \langle q, \gamma, q', \gamma', L \rangle \in \delta \),
- and similarly for all \( \langle q, \gamma, q', \gamma', R \rangle \in \delta \)

**The grounding of \( P \) is a Propositional Horn Logic program that is polynomial in the size of \( P \) (which is polynomial in the size of \( n \)).**

**ground(\( P \))** can be computed by a \textsc{LogSpace} transducer.
Exercise 5

Exercise. Show that the following property cannot be expressed in Datalog: The edge predicate has a proper cycle, i.e., a cycle that is not of the form edge\((a, a)\).
Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate \(\approx\) with the obvious semantics?
4. . . . an inequality predicate \(\not\approx\) with the obvious semantics?
Exercise 5

**Exercise.** Show that the following property cannot be expressed in Datalog: The edge predicate has a *proper* cycle, i.e., a cycle that is not of the form edge\((a, a)\).

Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate \(\approx\) with the obvious semantics?
4. . . . an inequality predicate \(\not\approx\) with the obvious semantics?

**Solution.**
Exercise 5

Exercise. Show that the following property cannot be expressed in Datalog: The edge predicate has a proper cycle, i.e., a cycle that is not of the form edge\((a, a)\).

Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate \(\approx\) with the obvious semantics?
4. . . . an inequality predicate \(\neq\) with the obvious semantics?

Solution.

0. We know that Datalog is homomorphism-closed, but the property of having a proper cycle is not, since any edge-cycle maps homomorphically onto edge\((a, a)\).
Exercise 5

**Exercise.** Show that the following property cannot be expressed in Datalog: The edge predicate has a *proper* cycle, i.e., a cycle that is not of the form \( \text{edge}(a, a) \).

Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate \( \approx \) with the obvious semantics?
4. . . . an inequality predicate \( \neq \) with the obvious semantics?

**Solution.**

0. We know that Datalog is *homomorphism-closed*, but the property of having a proper cycle is not, since any edge-cycle maps homomorphically onto \( \text{edge}(a, a) \).

1. 

\[
\begin{align*}
\langle(x, y) & \leftarrow \text{succ}(x, y) \quad \text{properEdge}(x, y) \leftarrow \text{edge}(x, y), \langle(x, y) \\
\langle(x, z) & \leftarrow \langle(x, y), \text{succ}(y, z) \quad \text{properEdge}(x, y) \leftarrow \text{edge}(x, y), \langle(y, x) \\
\text{properPath}(x, y) & \leftarrow \text{properEdge}(x, y) \\
\text{properPath}(x, z) & \leftarrow \text{properPath}(x, y), \text{properEdge}(y, z) \\
\text{properCycle}() & \leftarrow \text{properPath}(x, x)
\end{align*}
\]
Exercise 5

**Exercise.** Show that the following property cannot be expressed in Datalog: The edge predicate has a *proper* cycle, i.e., a cycle that is not of the form \(\text{edge}(a, a)\).

Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate \(\approx\) with the obvious semantics?
4. . . . an inequality predicate \(\neq\) with the obvious semantics?

**Solution.**

0. We know that Datalog is *homomorphism-closed*, but the property of having a proper cycle is not, since any edge-cycle maps homomorphically onto \(\text{edge}(a, a)\).

2. Suppose that \(P\) is a program entailing Accept iff edge contains a proper cycle.
Exercise 5

**Exercise.** Show that the following property cannot be expressed in Datalog: The edge predicate has a *proper* cycle, i.e., a cycle that is not of the form edge\((a, a)\).

Can you express this property using ... 

1. ... a successor ordering?
2. ... atomic EDB negation?
3. ... an equality predicate \(\approx\) with the obvious semantics?
4. ... an inequality predicate \(\neq\) with the obvious semantics?

**Solution.**

0. We know that Datalog is *homomorphism-closed*, but the property of having a proper cycle is not, since any edge-cycle maps homomorphically onto edge\((a, a)\).

2. ▶ Suppose that \(P\) is a program entailing Accept iff edge contains a proper cycle.
   ▶ Consider the DB \(I = \{\text{edge}(i,j) \mid 1 \leq i \leq j \leq 2\}\).
Exercise 5

Exercise. Show that the following property cannot be expressed in Datalog: The edge predicate has a proper cycle, i.e., a cycle that is not of the form edge(a, a).

Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate \( \approx \) with the obvious semantics?
4. . . . an inequality predicate \( \neq \) with the obvious semantics?

Solution.

0. We know that Datalog is homomorphism-closed, but the property of having a proper cycle is not, since any edge-cycle maps homomorphically onto edge(a, a).

2. ▶ Suppose that \( P \) is a program entailing Accept iff edge contains a proper cycle.
   ▶ Consider the DB \( I = \{ \text{edge}(i, j) \mid 1 \leq i \leq j \leq 2 \} \).
   ▶ Then \( P, I \models \text{Accept} \), and there must be a derivation of Accept that does not use negation.
Exercise 5

**Exercise.** Show that the following property cannot be expressed in Datalog: The edge predicate has a *proper* cycle, i.e., a cycle that is not of the form $\text{edge}(a, a)$.

Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate $\approx$ with the obvious semantics?
4. . . . an inequality predicate $\neq$ with the obvious semantics?

**Solution.**

0. We know that Datalog is *homomorphism-closed*, but the property of having a proper cycle is not, since any edge-cycle maps homomorphically onto $\text{edge}(a, a)$.

2. ▶ Suppose that $P$ is a program entailing Accept iff edge contains a proper cycle.
   ▶ Consider the DB $I = \{ \text{edge}(i, j) \mid 1 \leq i \leq j \leq 2 \}$.
   ▶ Then $P, I \models \text{Accept}$, and there must be a derivation of Accept that does not use negation.
   ▶ Let $P_+ \subseteq P$ be the negation-free subset of $P$. 
Exercise. Show that the following property cannot be expressed in Datalog: The edge predicate has a proper cycle, i.e., a cycle that is not of the form edge(a, a).

Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate ≈ with the obvious semantics?
4. . . . an inequality predicate ≠ with the obvious semantics?

Solution.

0. We know that Datalog is homomorphism-closed, but the property of having a proper cycle is not, since any edge-cycle maps homomorphically onto edge(a, a).

2. ▶ Suppose that P is a program entailing Accept iff edge contains a proper cycle.
   ▶ Consider the DB I = {edge(i, j) | 1 ≤ i ≤ j ≤ 2}.
   ▶ Then P, I |= Accept, and there must be a derivation of Accept that does not use negation.
   ▶ Let P⁺ ⊆ P be the negation-free subset of P.
   ▶ P⁺, I |= Accept, and I maps homomorphically onto {edge(a, a)}, contradiction.
Exercise 5

**Exercise.** Show that the following property cannot be expressed in Datalog: The edge predicate has a *proper* cycle, i.e., a cycle that is not of the form edge\((a, a)\).

Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate \(\approx\) with the obvious semantics?
4. . . . an inequality predicate \(\not{\approx}\) with the obvious semantics?

**Solution.**

0. We know that Datalog is *homomorphism-closed*, but the property of having a proper cycle is not, since any edge-cycle maps homomorphically onto edge\((a, a)\).

2. ▶ Suppose that \(P\) is a program entailing Accept iff edge contains a proper cycle.
   ▶ Consider the DB \(I = \{\text{edge}(i, j) \mid 1 \leq i \leq j \leq 2\}\).
   ▶ Then \(P, I \models \text{Accept}\), and there must be a derivation of Accept that does not use negation.
   ▶ Let \(P_+ \subseteq P\) be the negation-free subset of \(P\).
   ▶ \(P_+, I \models \text{Accept}\), and \(I\) maps homomorphically onto \(\{\text{edge}(a, a)\}\), contradiction.

3. Since \(\approx\) can be axiomatised using \(x \approx x \leftarrow\), Datalog with an equality predicate is not more expressive than Datalog.
Exercise 5

Exercise. Show that the following property cannot be expressed in Datalog: The edge predicate has a proper cycle, i.e., a cycle that is not of the form edge(a, a).

Can you express this property using . . .

1. . . . a successor ordering?
2. . . . atomic EDB negation?
3. . . . an equality predicate $\approx$ with the obvious semantics?
4. . . . an inequality predicate $\neq$ with the obvious semantics?

Solution.

0. We know that Datalog is homomorphism-closed, but the property of having a proper cycle is not, since any edge-cycle maps homomorphically onto edge(a, a).

2. ▶ Suppose that $P$ is a program entailing Accept iff edge contains a proper cycle.
   ▶ Consider the DB $I = \{ \text{edge}(i, j) \mid 1 \leq i \leq j \leq 2 \}$.
   ▶ Then $P, I \models \text{Accept}$, and there must be a derivation of Accept that does not use negation.
   ▶ Let $P_+ \subseteq P$ be the negation-free subset of $P$.
   ▶ $P_+, I \models \text{Accept}$, and $I$ maps homomorphically onto $\{ \text{edge}(a, a) \}$, contradiction.

3. Since $\approx$ can be axiomatised using $x \approx x \leftarrow$, Datalog with an equality predicate is not more expressive than Datalog.

4.

\[
\begin{align*}
\text{properEdge}(x, y) & \leftarrow \text{edge}(x, y) \land x \neq y \\
\text{properPath}(x, z) & \leftarrow \text{properPath}(x, y) \land \text{properEdge}(y, z) \\
\text{properCycle}() & \leftarrow \text{properPath}(x, x)
\end{align*}
\]