Exercise 9: Semi-Positive Datalog

Database Theory

2023-06-13

Maximilian Marx, Markus Krötzsch

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 ℓ-ary EDB predicate r occurring in P and all $1 \leq i \leq \ell$, we add a new rule $\mathrm{Top}(x_i) \leftarrow \mathrm{r}(x_1, \ldots, x_\ell)$,

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 ℓ-ary EDB predicate r occurring in P and all $1 \leq i \leq \ell$, we add a new rule $\mathrm{Top}(x_i) \leftarrow \mathrm{r}(x_1, \ldots, x_\ell)$,
  - for every constant c occurring in P, we add a new fact $\mathrm{Top}(c)$,

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 ℓ-ary EDB predicate r occurring in P and all $1 \leq i \leq \ell$, we add a new rule $\mathrm{Top}(x_i) \leftarrow \mathrm{r}(x_1, \ldots, x_\ell)$,
  - for every constant c occurring in P, we add a new fact $\mathrm{Top}(c)$,
  - for every rule $(H \leftarrow B)[x_1, \ldots, x_n] \in P$, we add the rule $H \leftarrow B \wedge \mathrm{Top}(x_1) \wedge \cdots \wedge \mathrm{Top}(x_\ell)$.

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 ℓ-ary EDB predicate r occurring in P and all $1 \leq i \leq \ell$, we add a new rule $\mathrm{Top}(x_i) \leftarrow \mathrm{r}(x_1, \ldots, x_\ell)$,
  - for every constant c occurring in P, we add a new fact $\mathrm{Top}(c)$,
  - for every rule $(H \leftarrow B)[x_1, \ldots, x_n] \in P$, we add the rule $H \leftarrow B \wedge \mathrm{Top}(x_1) \wedge \cdots \wedge \mathrm{Top}(x_\ell)$.
- Then for every fact φ over the signature of P, we have that P' entails φ over an instance D iff P entails φ 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 ℓ-ary EDB predicate r occurring in P and all $1 \leq i \leq \ell$, we add a new rule $\mathrm{Top}(x_i) \leftarrow \mathrm{r}(x_1, \ldots, x_\ell)$,
  - for every constant c occurring in P, we add a new fact $\mathrm{Top}(c)$,
  - for every rule $(H \leftarrow B)[x_1, \ldots, x_n] \in P$, we add the rule $H \leftarrow B \wedge \mathrm{Top}(x_1) \wedge \cdots \wedge \mathrm{Top}(x_\ell)$.
- Then for every fact φ over the signature of P, we have that P' entails φ over an instance D iff P entails φ 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.

$$
\begin{aligned}
\mathrm{Odd}(x) &\leftarrow \mathrm{first}(x) \\
\mathrm{Odd}(y) &\leftarrow \mathrm{Even}(x), \mathrm{succ}(x, y) \\
\mathrm{Even}(y) &\leftarrow \mathrm{Odd}(x), \mathrm{succ}(x, y) \\
\mathrm{EvenParity}() &\leftarrow \mathrm{Even}(x), \mathrm{last}(x)
\end{aligned}
$$

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.

$$
\begin{aligned}
<(x, y) &\leftarrow \mathrm{succ}(x, y) \\
<(x, z) &\leftarrow <(x, y), \mathrm{succ}(y, z) \\
<>(x, y), <>(y, x) &\leftarrow <(x, y) \\
\mathrm{TwoOutgoingEdges}() &\leftarrow \mathrm{edge}(x, y), \mathrm{edge}(x, z), <>(y, z)
\end{aligned}
$$

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.

$$
\begin{aligned}
<(x, y) &\leftarrow \text{succ}(x, y) \\
<(x, z) &\leftarrow <(x, y), \text{succ}(y, z) \\
<>(x, y), <>(y, x) &\leftarrow <(x, y) \\
\text{TwoOutgoingEdges}() &\leftarrow \text{edge}(x, y), \text{edge}(x, z), <>(y, z)
\end{aligned}
$$

3. This is (most likely) not expressible (unless $\mathrm{P} = \mathrm{NP}$), 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. $\mathrm{D}(x, y, k)$ x and y are not reachable via a path of length at most k

   $\mathrm{N}(x, y, z, k)$ there is no path of length $k + 1$ from x to z via y

$$\mathrm{D}(x, y, \ell), \mathrm{D}(y, x, \ell) \leftarrow \neg\mathrm{edge}(x, y), \mathrm{first}(\ell), <>(x, y)$$

$$\mathrm{N}(x, y, z, k) \leftarrow \mathrm{first}(y), \mathrm{D}(x, y, k)$$

$$\mathrm{N}(x, y', z, k) \leftarrow \mathrm{succ}(y, y'), \mathrm{N}(x, y, z, k), \mathrm{D}(x, y', k)$$

$$\mathrm{D}(x, z, k') \leftarrow \mathrm{succ}(k, k'), \mathrm{D}(x, z, k), \mathrm{last}(y), \mathrm{N}(x, y, z, k)$$

$$\mathrm{N}(x, y, z, k) \leftarrow \mathrm{first}(y), \mathrm{D}(y, z, k)$$

$$\mathrm{N}(x, y', z, k) \leftarrow \mathrm{succ}(y, y'), \mathrm{N}(x, y, z, k), \mathrm{D}(y', z, k)$$

$$\mathrm{Ans}() \leftarrow \mathrm{D}(x, y, k), \mathrm{last}(k)$$

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.

5.

$$\mathrm{oneEdge}(x,y) \leftarrow \mathrm{first}(y), \mathrm{edge}(x,y)$$
$$\mathrm{oneEdge}(x,z) \leftarrow \mathrm{noEdge}(x,y), \mathrm{succ}(y,z), \mathrm{edge}(x,z)$$
$$\mathrm{oneEdge}(x,z) \leftarrow \mathrm{oneEdge}(x,y), \mathrm{succ}(y,z), \neg\mathrm{edge}(x,z)$$
$$\mathrm{r}(x) \leftarrow \mathrm{last}(y), \mathrm{noEdge}(x,y)$$
$$\mathrm{r}(x) \leftarrow \mathrm{last}(y), \mathrm{oneEdge}(x,y)$$
$$\mathrm{NoTwoOutEdges}() \leftarrow \mathrm{s}(x), \mathrm{last}(x)$$

$$\mathrm{noEdge}(x,y) \leftarrow \mathrm{first}(y), \neg\mathrm{edge}(x,y)$$
$$\mathrm{noEdge}(x,z) \leftarrow \mathrm{noEdge}(x,y), \mathrm{succ}(y,z), \neg\mathrm{edge}(x,z)$$
$$\mathrm{s}(x) \leftarrow \mathrm{first}(x), \mathrm{r}(x)$$
$$\mathrm{s}(y) \leftarrow \mathrm{succ}(x,y), \mathrm{s}(x), \mathrm{r}(y)$$

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.

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

$$\text{Conn}(x), \text{Reachable}(x) \leftarrow \text{first}(x) \qquad \text{Reachable}(x) \leftarrow \text{Reachable}(x), \text{succ}(x, y), \text{Conn}(y)$$

$$\text{Conn}(y) \leftarrow \text{Conn}(x), \text{edge}(x, y) \qquad \text{Conn}(y) \leftarrow \text{Conn}(x), \text{edge}(y, x)$$

$$\text{Connected}() \leftarrow \text{last}(x), \text{Reachable}(x)$$

$$\text{NoInEdge}(x, y) \leftarrow \text{first}(x), \neg\text{edge}(x, y)$$

$$\text{NoOutEdge}(x, y) \leftarrow \text{first}(x), \neg\text{edge}(y, x)$$

$$\text{NoInEdge}(x', y) \leftarrow \text{succ}(x, x'), \text{NoInEdge}(x, y), \neg\text{edge}(x', y)$$

$$\text{NoOutEdge}(x', y) \leftarrow \text{succ}(x, x'), \text{NoOutEdge}(x, y), \neg\text{edge}(y, x')$$

$$\text{NoCycle}() \leftarrow \text{last}(x), \text{NoInEdge(x, y)}, \text{NoOutEdge(x, z)}$$

with NoTwoOutEdges() defined as in 5., and NoTwoInEdges() defined analogously.

Exercise 3

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

Exercise 3

Exercise. A Horn logic program is in *propHorn2* if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \wedge B_2$.
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 $H \leftarrow B_1 \wedge B_2$.
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 $H \leftarrow B_1 \wedge B_2$.
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 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 $H \leftarrow B_1 \wedge B_2$.
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 $B \wedge B \rightarrow H \in P'$,

Exercise 3

Exercise. A Horn logic program is in *propHorn2* if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \wedge B_2$.
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 $B \wedge B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \wedge \ldots \wedge B_n$ with $n \geq 3$, then add $B_1 \wedge B_2 \rightarrow F_2, F_2 \wedge B_3 \rightarrow F_3, \ldots, F_{n-1} \wedge B_n \rightarrow H \in P'$ 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 $H \leftarrow B_1 \wedge B_2$.
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 $B \wedge B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \wedge \ldots \wedge B_n$ with $n \geq 3$, then add $B_1 \wedge B_2 \rightarrow F_2, F_2 \wedge B_3 \rightarrow F_3, \ldots, F_{n-1} \wedge B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - otherwise, add $Body \rightarrow H \in P'$.

Exercise 3

Exercise. A Horn logic program is in *propHorn2* if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \wedge B_2$.
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 Body $\rightarrow H \in P$,
  - if Body $= B$ consists of a single atom, then add $B \wedge B \rightarrow H \in P'$,
  - if Body $= B_1 \wedge \ldots \wedge B_n$ with $n \geq 3$, then add $B_1 \wedge B_2 \rightarrow F_2, F_2 \wedge B_3 \rightarrow F_3, \ldots, F_{n-1} \wedge B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - otherwise, add $Body \rightarrow H \in P'$.
- 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 *propHorn2* if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \wedge B_2$.
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 $B \wedge B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \wedge \ldots \wedge B_n$ with $n \geq 3$, then add $B_1 \wedge B_2 \rightarrow F_2, F_2 \wedge B_3 \rightarrow F_3, \ldots, F_{n-1} \wedge B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - otherwise, add $\mathit{Body} \rightarrow H \in P'$.
- 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 $H \leftarrow B_1 \wedge B_2$.
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 Body $\rightarrow H \in P$,
  - if Body $= B$ consists of a single atom, then add $B \wedge B \rightarrow H \in P'$,
  - if Body $= B_1 \wedge \ldots \wedge B_n$ with $n \geq 3$, then add $B_1 \wedge B_2 \rightarrow F_2, F_2 \wedge B_3 \rightarrow F_3, \ldots, F_{n-1} \wedge B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - otherwise, add $Body \rightarrow H \in P'$.
- 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:
  - count number of body atoms to generate these rules

Exercise 3

Exercise. A Horn logic program is in *propHorn2* if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \wedge B_2$.
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 $B \wedge B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \wedge \ldots \wedge B_n$ with $n \geq 3$, then add $B_1 \wedge B_2 \rightarrow F_2, F_2 \wedge B_3 \rightarrow F_3, \ldots, F_{n-1} \wedge B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - otherwise, add $\mathit{Body} \rightarrow H \in P'$.
- 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:
  - 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 *propHorn2* if every rule it contains is of the form $H \leftarrow$ or $H \leftarrow B_1 \wedge B_2$.
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 $B \wedge B \rightarrow H \in P'$,
  - if $\text{Body} = B_1 \wedge \ldots \wedge B_n$ with $n \geq 3$, then add $B_1 \wedge B_2 \rightarrow F_2, F_2 \wedge B_3 \rightarrow F_3, \ldots, F_{n-1} \wedge B_n \rightarrow H \in P'$ where $F_2, \ldots, F_{n-1}$ are fresh propositional variables.
  - otherwise, add $Body \rightarrow H \in P'$.
- 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:
  - 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 length 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 $\mathcal{M} = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle$ 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 $\mathcal{M} = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta \rangle$ and an input word $w = w_1, \ldots, w_n \in \Sigma^*$.
- We define a Datalog program P such that P entails a predicate $Accept()$ iff the TM $\mathcal{M}$ accepts w.

Exercise 4

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 $\mathit{Accept}()$ iff the TM $\mathcal{M}$ accepts w.
- Since $\mathcal{M}$ is polynomial, $\mathcal{M}$ halts after at most n^k steps for some $k \geq 0$.

Exercise 4

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

Exercise 4

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 $Accept()$ 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:
  - $\mathrm{cell}_{i,j}$ for all $1 \leq i \leq j \leq n^k + 1$, and
  - all elements $q \in Q$ and $\gamma \in \Gamma$
- Facts:
  - $\mathrm{right}(\mathrm{cell}_{i,j}, \mathrm{cell}_{i,j+1}), \mathrm{future}(\mathrm{cell}_{i,j}, \mathrm{cell}_{i+1,j})$, for $1 \leq i \leq j \leq n^k$,
  - $\mathrm{S}(\mathrm{cell}_{0,i}, w_i)$ for $1 \leq i \leq n$, and $\mathrm{S}(\mathrm{cell}_{0,i}, b)$ for $n + 1 \leq i \leq n^k$, and
  - $\mathrm{T}(\mathrm{cell}_{0,0}, q_0)$

Exercise 4

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 $Accept()$ 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:
  - $\mathrm{cell}_{i,j}$ for all $1 \leq i \leq j \leq n^k + 1$, and
  - all elements $q \in Q$ and $\gamma \in \Gamma$
- Facts:
  - $\mathrm{right}(\mathrm{cell}_{i,j}, \mathrm{cell}_{i,j+1}), \mathrm{future}(\mathrm{cell}_{i,j}, \mathrm{cell}_{i+1,j})$, for $1 \leq i \leq j \leq n^k$,
  - $\mathrm{S}(\mathrm{cell}_{0,i}, w_i)$ for $1 \leq i \leq n$, and $\mathrm{S}(\mathrm{cell}_{0,i}, b)$ for $n + 1 \leq i \leq n^k$, and
  - $\mathrm{T}(\mathrm{cell}_{0,0}, q_0)$
- Rules:
  - $\mathrm{Accept}() \leftarrow \mathrm{T}(x, q_f)$,
  - $\mathrm{NTR}(z) \leftarrow \mathrm{T}(x, y) \wedge \mathrm{right}(x, z)$, $\mathrm{NTR}(y) \leftarrow \mathrm{NTR}(x) \wedge \mathrm{right}(x, y)$, $\mathrm{NTL}(z) \leftarrow \mathrm{T}(x, y) \wedge \mathrm{right}(z, x)$, $\mathrm{NTL}(x) \leftarrow \mathrm{right}(\mathrm{x}, \mathrm{y}) \wedge \mathrm{NTL}(y)$, $\mathrm{NT}(x) \leftarrow \mathrm{NTR}(x)$, $\mathrm{NT}(x) \leftarrow \mathrm{NTL}(x)$, $\mathrm{S}(y, z) \leftarrow \mathrm{NT}(x) \wedge \mathrm{future}(x, y) \wedge \mathrm{S}(x, z)$,
  - $T(z, q') \leftarrow T(x, q) \wedge S(x, \gamma) \wedge \mathrm{future}(x, y) \wedge \mathrm{right}(z, y)$, $S(y, \gamma') \leftarrow T(x, q) \wedge S(x, \gamma) \wedge \mathrm{future}(x, y)$ 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 $\mathcal{M} = \langle Q, \Gamma, \Sigma, q_0, q_f, \delta\rangle$ and an input word $w = w_1, \ldots, w_n \in \Sigma^*$.
- We define a Datalog program P such that P entails a predicate $Accept()$ 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:
  - $\mathrm{cell}_{i,j}$ for all $1 \leq i \leq j \leq n^k + 1$, and
  - all elements $q \in Q$ and $\gamma \in \Gamma$
- Facts:
  - $\mathrm{right}(\mathrm{cell}_{i,j}, \mathrm{cell}_{i,j+1}), \mathrm{future}(\mathrm{cell}_{i,j}, \mathrm{cell}_{i+1,j})$, for $1 \leq i \leq j \leq n^k$,
  - $\mathrm{S}(\mathrm{cell}_{0,i}, w_i)$ for $1 \leq i \leq n$, and $\mathrm{S}(\mathrm{cell}_{0,i}, b)$ for $n + 1 \leq i \leq n^k$, and
  - $\mathrm{T}(\mathrm{cell}_{0,0}, q_0)$
- Rules:
  - $\mathrm{Accept}() \leftarrow \mathrm{T}(x, q_f)$,
  - $\mathrm{NTR}(z) \leftarrow \mathrm{T}(x, y) \wedge \mathrm{right}(x, z)$, $\mathrm{NTR}(y) \leftarrow \mathrm{NTR}(x) \wedge \mathrm{right}(x, y)$, $\mathrm{NTL}(z) \leftarrow \mathrm{T}(x, y) \wedge \mathrm{right}(z, x)$, $\mathrm{NTL}(x) \leftarrow \mathrm{right(x, y)} \wedge \mathrm{NTL}(y)$, $\mathrm{NT}(x) \leftarrow \mathrm{NTR}(x)$, $\mathrm{NT}(x) \leftarrow \mathrm{NTL}(x)$, $\mathrm{S}(y, z) \leftarrow \mathrm{NT}(x) \wedge \mathrm{future}(x, y) \wedge \mathrm{S}(x, z)$,
  - $T(z, q') \leftarrow T(x, q) \wedge S(x, \gamma) \wedge \mathrm{future}(x, y) \wedge \mathrm{right}(z, y)$, $S(y, \gamma') \leftarrow T(x, q) \wedge S(x, \gamma) \wedge \mathrm{future}(x, y)$ 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 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 $Accept()$ 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:
  - $\mathrm{cell}_{i,j}$ for all $1 \leq i \leq j \leq n^k + 1$, and
  - all elements $q \in Q$ and $\gamma \in \Gamma$
- Facts:
  - $\mathrm{right}(\mathrm{cell}_{i,j}, \mathrm{cell}_{i,j+1}), \mathrm{future}(\mathrm{cell}_{i,j}, \mathrm{cell}_{i+1,j})$, for $1 \leq i \leq j \leq n^k$,
  - $\mathrm{S}(\mathrm{cell}_{0,i}, w_i)$ for $1 \leq i \leq n$, and $\mathrm{S}(\mathrm{cell}_{0,i}, b)$ for $n + 1 \leq i \leq n^k$, and
  - $\mathrm{T}(\mathrm{cell}_{0,0}, q_0)$
- Rules:
  - $\mathrm{Accept}() \leftarrow \mathrm{T}(x, q_f)$,
  - $\mathrm{NTR}(z) \leftarrow \mathrm{T}(x, y) \wedge \mathrm{right}(x, z)$, $\mathrm{NTR}(y) \leftarrow \mathrm{NTR}(x) \wedge \mathrm{right}(x, y)$, $\mathrm{NTL}(z) \leftarrow \mathrm{T}(x, y) \wedge \mathrm{right}(z, x)$, $\mathrm{NTL}(x) \leftarrow \mathrm{right}(x, y) \wedge \mathrm{NTL}(y)$, $\mathrm{NT}(x) \leftarrow \mathrm{NTR}(x)$, $\mathrm{NT}(x) \leftarrow \mathrm{NTL}(x)$, $\mathrm{S}(y, z) \leftarrow \mathrm{NT}(x) \wedge \mathrm{future}(x, y) \wedge \mathrm{S}(x, z)$,
  - $T(z, q') \leftarrow T(x, q) \wedge S(x, \gamma) \wedge \mathrm{future}(x, y) \wedge \mathrm{right}(z, y)$, $S(y, \gamma') \leftarrow T(x, q) \wedge S(x, \gamma) \wedge \mathrm{future}(x, y)$ 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).
- $\mathrm{ground}(P)$ 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 $\not\approx$ 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).

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).
1.

$$
\begin{aligned}
<(x, y) &\leftarrow \mathrm{succ}(x, y) \\
<(x, z) &\leftarrow <(x, y), \mathrm{succ}(y, z) \\
\mathrm{properPath}(x, y) &\leftarrow \mathrm{properEdge}(x, y) \\
\mathrm{properPath}(x, z) &\leftarrow \mathrm{properPath}(x, y), \mathrm{properEdge}(y, z) \\
\mathrm{properCycle}() &\leftarrow \mathrm{properPath}(x, x)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{properEdge}(x, y) &\leftarrow \mathrm{edge}(x, y), <(x, y) \\
\mathrm{properEdge}(x, y) &\leftarrow \mathrm{edge}(x, y), <(y, x)
\end{aligned}
$$

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.

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 $\mathcal{I} = \{ \text{edge}(i, j) \mid i, j \in \{ 1, 2 \} \}$.

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

$$\text{properEdge}(x, y) \leftarrow \text{edge}(x, y) \wedge x \not\approx y \qquad \text{properPath}(x, y) \leftarrow \text{properEdge}(x, y)$$

$$\text{properPath}(x, z) \leftarrow \text{properPath}(x, y) \wedge \text{properEdge}(y, z) \qquad \text{properCycle}() \leftarrow \text{properPath}(x, x)$$