Exercise 4: Conjunctive Queries, CSP, and Hypergraphs

Database Theory

2022-05-03

Maximilian Marx, Markus Krötzsch

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \land r(y, z) \land r(z, v) \land s(x, y, z) \land s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \land s(x, z, v) \land r(u, z) \land t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E\rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

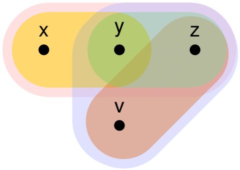
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

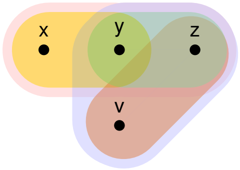
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



(1) delete $\langle x, y \rangle$

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

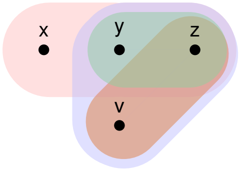
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



(1) delete $\langle x, y \rangle$

(1) delete $\langle y, z \rangle$

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

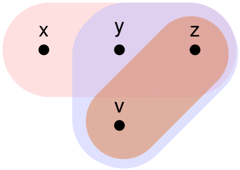
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



(1) delete $\langle x, y \rangle$

(1) delete $\langle y, z \rangle$

(1) delete $\langle z, v \rangle$

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

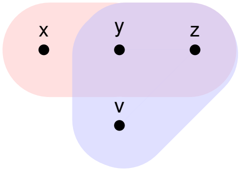
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



(1) delete $\langle x, y \rangle$

(1) delete $\langle y, z \rangle$

(1) delete $\langle z, v \rangle$

(2) delete x

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

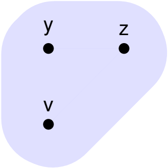
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



(1) delete $\langle x, y \rangle$

(1) delete $\langle y, z \rangle$

(1) delete $\langle z, v \rangle$

(2) delete x

(2) delete y, z, v

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \land r(y, z) \land r(z, v) \land s(x, y, z) \land s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \land s(x, z, v) \land r(u, z) \land t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.

(1) delete $\langle x, y \rangle$

(1) delete $\langle y, z \rangle$

(1) delete $\langle z, v \rangle$

(2) delete x

(2) delete y, z, v

$\rightsquigarrow$ query is acyclic.

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

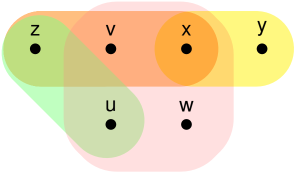
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

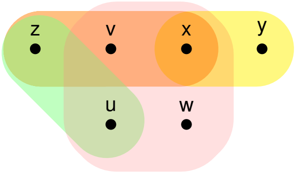
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



(2) delete w

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

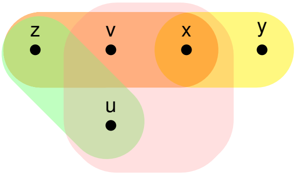
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



(2) delete w

(2) delete y

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

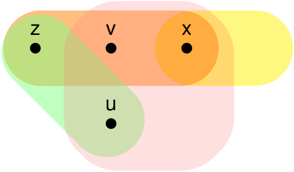
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



(2) delete w

(2) delete y

(1) delete $\langle x \rangle$

Exercise 1

Exercise. Decide if the following conjunctive queries are tree queries by applying (one version of) the GYO algorithm.

1. $\exists x, y, z, v.\ r(x, y) \wedge r(y, z) \wedge r(z, v) \wedge s(x, y, z) \wedge s(y, z, v)$
2. $\exists x, y, z, u, v, w.\ r(x, y) \wedge s(x, z, v) \wedge r(u, z) \wedge t(x, v, u, w)$

Definition (Lecture 6, Slides 24)

A GYO-reduction is a procedure to check acyclicity:

- Input: hypergraph $H = \langle V, E \rangle$ (we do not need relation labels here)
- Output: GYO-reduct of H

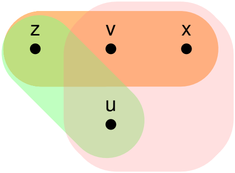
Apply the following simplification rules as long as possible:

(1) Delete all hyperedges that are empty or that are contained in other hyperedges.

(2) Delete all vertices that occur in at most one hyperedge.

The input query is a tree query if the GYO-reduct of its hypergraph is the empty hypergraph.

Solution.



(2) delete w

(2) delete y

(1) delete $\langle x \rangle$

$\rightsquigarrow$ query is not acyclic.

Exercise 2

Exercise.
It was outlined in the lecture how to eliminate constants from CQs to transform them to graphs. Apply this transformation to the following query:

$$\exists x, y, z.\ \text{mother}(x, y) \land \text{father}(x, z) \land \text{bornIn}(y, \texttt{"Dresden"}) \land \text{bornIn}(z, \texttt{"Dresden"}).$$

Is the transformed query a tree query? Imagine we would keep constants in the hypergraph, so that each hypergraph would contain two kinds of vertices (variables and constants). How could we modify the GYO algorithm to handle such hypergraphs directly?

Exercise 2

Exercise.
It was outlined in the lecture how to eliminate constants from CQs to transform them to graphs. Apply this transformation to the following query:

$$\exists x, y, z.\ \mathsf{mother}(x, y) \wedge \mathsf{father}(x, z) \wedge \mathsf{bornIn}(y, \texttt{"Dresden"}) \wedge \mathsf{bornIn}(z, \texttt{"Dresden"}).$$

Is the transformed query a tree query? Imagine we would keep constants in the hypergraph, so that each hypergraph would contain two kinds of vertices (variables and constants). How could we modify the GYO algorithm to handle such hypergraphs directly?
Solution.

Exercise 2

Exercise.
It was outlined in the lecture how to eliminate constants from CQs to transform them to graphs. Apply this transformation to the following query:

$$\exists x, y, z.\ \text{mother}(x, y) \wedge \text{father}(x, z) \wedge \text{bornIn}(y, \texttt{"Dresden"}) \wedge \text{bornIn}(z, \texttt{"Dresden"}).$$

Is the transformed query a tree query? Imagine we would keep constants in the hypergraph, so that each hypergraph would contain two kinds of vertices (variables and constants). How could we modify the GYO algorithm to handle such hypergraphs directly?

Solution.

- $\exists x, y, z \quad .\ \text{mother}(x, y) \wedge \text{father}(x, z) \wedge \text{bornIn}(y, \texttt{"Dresden"}) \wedge \text{bornIn}(z, \texttt{"Dresden"})$

Exercise 2

Exercise.
It was outlined in the lecture how to eliminate constants from CQs to transform them to graphs. Apply this transformation to the following query:

$$\exists x, y, z.\ \mathrm{mother}(x, y) \wedge \mathrm{father}(x, z) \wedge \mathrm{bornIn}(y, \texttt{"Dresden"}) \wedge \mathrm{bornIn}(z, \texttt{"Dresden"}).$$

Is the transformed query a tree query? Imagine we would keep constants in the hypergraph, so that each hypergraph would contain two kinds of vertices (variables and constants). How could we modify the GYO algorithm to handle such hypergraphs directly?

Solution.

- $\exists x, y, z, v\quad .\ \mathrm{mother}(x, y) \wedge \mathrm{father}(x, z) \wedge \mathrm{bornIn}(y, v) \wedge \mathrm{R}_{\texttt{"Dresden"}}(v) \wedge \mathrm{bornIn}(z, \texttt{"Dresden"})$

Exercise 2

Exercise.
It was outlined in the lecture how to eliminate constants from CQs to transform them to graphs. Apply this transformation to the following query:

$$\exists x, y, z.\ \mathrm{mother}(x, y) \land \mathrm{father}(x, z) \land \mathrm{bornIn}(y, \texttt{"Dresden"}) \land \mathrm{bornIn}(z, \texttt{"Dresden"}).$$

Is the transformed query a tree query? Imagine we would keep constants in the hypergraph, so that each hypergraph would contain two kinds of vertices (variables and constants). How could we modify the GYO algorithm to handle such hypergraphs directly?
Solution.

- $\exists x, y, z, v, w.\ \mathrm{mother}(x, y) \land \mathrm{father}(x, z) \land \mathrm{bornIn}(y, v) \land \mathrm{R}_{\texttt{"Dresden"}}(v) \land \mathrm{bornIn}(z, w) \land \mathrm{R}_{\texttt{"Dresden"}}(w)$

Exercise 2

Exercise.
It was outlined in the lecture how to eliminate constants from CQs to transform them to graphs. Apply this transformation to the following query:

$$\exists x, y, z.\ \text{mother}(x, y) \wedge \text{father}(x, z) \wedge \text{bornIn}(y, \texttt{"Dresden"}) \wedge \text{bornIn}(z, \texttt{"Dresden"}).$$

Is the transformed query a tree query? Imagine we would keep constants in the hypergraph, so that each hypergraph would contain two kinds of vertices (variables and constants). How could we modify the GYO algorithm to handle such hypergraphs directly?

Solution.

- $\exists x, y, z, v, w.\ \text{mother}(x, y) \wedge \text{father}(x, z) \wedge \text{bornIn}(y, v) \wedge \text{R}_{\texttt{"Dresden"}}(v) \wedge \text{bornIn}(z, w) \wedge \text{R}_{\texttt{"Dresden"}}(w)$
- The query is acyclic.

Exercise 2

Exercise.
It was outlined in the lecture how to eliminate constants from CQs to transform them to graphs. Apply this transformation to the following query:

$$\exists x, y, z.\ \text{mother}(x, y) \wedge \text{father}(x, z) \wedge \text{bornIn}(y, \texttt{"Dresden"}) \wedge \text{bornIn}(z, \texttt{"Dresden"}).$$

Is the transformed query a tree query? Imagine we would keep constants in the hypergraph, so that each hypergraph would contain two kinds of vertices (variables and constants). How could we modify the GYO algorithm to handle such hypergraphs directly?
Solution.

- $\exists x, y, z, v, w.\ \text{mother}(x, y) \wedge \text{father}(x, z) \wedge \text{bornIn}(y, v) \wedge \text{R}_{\texttt{"Dresden"}}(v) \wedge \text{bornIn}(z, w) \wedge \text{R}_{\texttt{"Dresden"}}(w)$
- The query is acyclic.
- Add rule: “Delete all vertices labelled with constants.”

Exercise 3

Exercise. Solve the following combinatorial crossword puzzle using Yannakakis' algorithm (in spirit). Specify the join tree that you are using.

x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_8	■	x_9	■	■	■	x_{10}
x_{11}	■	x_{12}	■	x_{13}	x_{14}	x_{15}
x_{16}	■	x_{17}	■	■	■	x_{18}
x_{19}	■	x_{20}	■	x_{21}	x_{22}	x_{23}

1 hor.:

B	R	I	S	T	O	L
C	A	R	A	M	E	L
P	H	A	R	A	O	H
S	P	I	N	A	C	H
T	S	U	N	A	M	I

1 vert.:

C	L	E	A	R
H	U	M	A	N
P	E	A	C	E
S	H	A	R	K
T	I	G	E	R

3 vert.:

H	A	P	P	Y
I	N	F	E	R
L	A	B	O	R
L	A	T	E	R
U	N	T	I	L

7 vert.:

H	E	A	R	T
H	O	N	E	Y
I	R	O	N	Y
L	O	G	I	C
M	A	G	I	C

13 hor.:

A	N	D
C	A	T
D	I	M
L	A	G
W	I	N

21 hor.:

A	R	C
F	E	E
L	O	W
T	W	O
W	A	Y

Exercise 3

Exercise. Solve the following combinatorial crossword puzzle using Yannakakis' algorithm (in spirit). Specify the join tree that you are using.

Solution.

x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_8	■	x_9	■	■	■	x_{10}
x_{11}	■	x_{12}	■	x_{13}	x_{14}	x_{15}
x_{16}	■	x_{17}	■	■	■	x_{18}
x_{19}	■	x_{20}	■	x_{21}	x_{22}	x_{23}

1 hor.:

B	R	I	S	T	O	L
C	A	R	A	M	E	L
P	H	A	R	A	O	H
S	P	I	N	A	C	H
T	S	U	N	A	M	I

1 vert.:

C	L	E	A	R
H	U	M	A	N
P	E	A	C	E
S	H	A	R	K
T	I	G	E	R

3 vert.:

H	A	P	P	Y
I	N	F	E	R
L	A	B	O	R
L	A	T	E	R
U	N	T	I	L

7 vert.:

H	E	A	R	T
H	O	N	E	Y
I	R	O	N	Y
L	O	G	I	C
M	A	G	I	C

13 hor.:

A	N	D
C	A	T
D	I	M
L	A	G
W	I	N

21 hor.:

A	R	C
F	E	E
L	O	W
T	W	O
W	A	Y

Exercise 3

Exercise. Solve the following combinatorial crossword puzzle using Yannakakis' algorithm (in spirit). Specify the join tree that you are using.

x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_8	■	x_9	■	■	■	x_{10}
x_{11}	■	x_{12}	■	x_{13}	x_{14}	x_{15}
x_{16}	■	x_{17}	■	■	■	x_{18}
x_{19}	■	x_{20}	■	x_{21}	x_{22}	x_{23}

1 hor.:

B	*R*	*I*	*S*	*T*	*O*	*L*
C	*A*	*R*	*A*	*M*	*E*	*L*
P	*H*	*A*	*R*	*A*	*O*	*H*
S	*P*	*I*	*N*	*A*	*C*	*H*
T	*S*	*U*	*N*	*A*	*M*	*I*

1 vert.:

C	*L*	*E*	*A*	*R*
H	*U*	*M*	*A*	*N*
P	*E*	*A*	*C*	*E*
S	*H*	*A*	*R*	*K*
T	*I*	*G*	*E*	*R*

3 vert.:

H	*A*	*P*	*P*	*Y*
I	*N*	*F*	*E*	*R*
L	*A*	*B*	*O*	*R*
L	*A*	*T*	*E*	*R*
U	*N*	*T*	*I*	*L*

7 vert.:

H	*E*	*A*	*R*	*T*
H	*O*	*N*	*E*	*Y*
I	*R*	*O*	*N*	*Y*
L	*O*	*G*	*I*	*C*
M	*A*	*G*	*I*	*C*

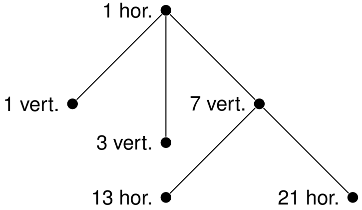
13 hor.:

A	*N*	*D*
C	*A*	*T*
D	*I*	*M*
L	*A*	*G*
W	*I*	*N*

21 hor.:

A	*R*	*C*
F	*E*	*E*
L	*O*	*W*
T	*W*	*O*
W	*A*	*Y*

Solution.
Join tree:



Exercise 3

Exercise. Solve the following combinatorial crossword puzzle using Yannakakis' algorithm (in spirit). Specify the join tree that you are using.

x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_8		x_9				x_{10}
x_{11}		x_{12}		x_{13}	x_{14}	x_{15}
x_{16}		x_{17}				x_{18}
x_{19}		x_{20}		x_{21}	x_{22}	x_{23}

1 hor.:

B	R	I	S	T	O	L
C	A	R	A	M	E	L
P	H	A	R	A	O	H
S	P	I	N	A	C	H
T	S	U	N	A	M	I

1 vert.:

C	L	E	A	R
H	U	M	A	N
P	E	A	C	E
S	H	A	R	K
T	I	G	E	R

3 vert.:

H	A	P	P	Y
I	N	F	E	R
L	A	B	O	R
L	A	T	E	R
U	N	T	I	L

7 vert.:

H	E	A	R	T
H	O	N	E	Y
I	R	O	N	Y
L	O	G	I	C
M	A	G	I	C

13 hor.:

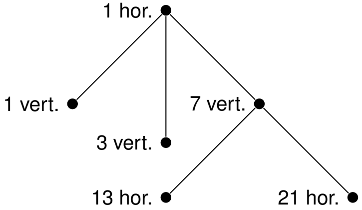
A	N	D
C	A	T
D	I	M
L	A	G
W	I	N

21 hor.:

A	R	C
F	E	E
L	O	W
T	W	O
W	A	Y

Solution.

Join tree:



S	P	I	N	A	C	H
H		N				O
A		F		W	I	N
R		E				E
K		R		W	A	Y

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.
Solution.

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.
Solution. Let us consider a naive strategy:

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.
Solution. Let us consider a naive strategy:

- Let $q = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w)$

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.
Solution. Let us consider a naive strategy:

- Let $q = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w)$
- Let $q' = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w) \wedge \mathrm{X}(v, w, x, y, z)$

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.

Solution. Let us consider a naive strategy:

- Let $q = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w)$
- Let $q' = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w) \wedge \mathrm{X}(v, w, x, y, z)$
- For a database instance $\mathcal{I}$, let $\mathcal{I}' = \mathcal{I} \cup \{X(c_1, \ldots, c_5) \mid c_1, \ldots, c_5 \in \mathbf{adom}(\mathcal{I})\}$.

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.
Solution. Let us consider a naive strategy:

- Let $q = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w)$
- Let $q' = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w) \wedge \mathrm{X}(v, w, x, y, z)$
- For a database instance $\mathcal{I}$, let $\mathcal{I}' = \mathcal{I} \cup \{X(c_1, \ldots, c_5) \mid c_1, \ldots, c_5 \in \mathbf{adom}(\mathcal{I})\}$.
- Then $\mathcal{I} \models q$ iff $\mathcal{I}' \models q'$.

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.

Solution. Let us consider a naive strategy:

- Let $q = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w)$
- Let $q' = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w) \wedge \mathrm{X}(v, w, x, y, z)$
- For a database instance $\mathcal{I}$, let $\mathcal{I}' = \mathcal{I} \cup \{X(c_1, \ldots, c_5) \mid c_1, \ldots, c_5 \in \mathbf{adom}(\mathcal{I})\}$.
- Then $\mathcal{I} \models q$ iff $\mathcal{I}' \models q'$.
- Query q' is acyclic and can thus be answered in polynomial time with respect to the size of the database $\mathcal{I}'$.

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.

Solution. Let us consider a naive strategy:

- Let $q = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w)$
- Let $q' = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w) \wedge \mathrm{X}(v, w, x, y, z)$
- For a database instance $\mathcal{I}$, let $\mathcal{I}' = \mathcal{I} \cup \{X(c_1, \ldots, c_5) \mid c_1, \ldots, c_5 \in \textbf{adom}(\mathcal{I})\}$.
- Then $\mathcal{I} \models q$ iff $\mathcal{I}' \models q'$.
- Query q' is acyclic and can thus be answered in polynomial time with respect to the size of the database $\mathcal{I}'$.
- But computing $\mathcal{I}'$ from $\mathcal{I}$ requires adding $|\textbf{adom}(\mathcal{I})|^5$ new facts.

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.
Solution. Let us consider a naive strategy:

- Let $q = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w)$
- Let $q' = \exists v, w, x, y, z.\ \mathrm{P}(x, y, z) \wedge \mathrm{Q}(z, w) \wedge \mathrm{R}(w, v, x) \wedge \mathrm{S}(y, w) \wedge \mathrm{X}(v, w, x, y, z)$
- For a database instance $\mathcal{I}$, let $\mathcal{I}' = \mathcal{I} \cup \{X(c_1, \ldots, c_5) \mid c_1, \ldots, c_5 \in \mathbf{adom}(\mathcal{I})\}$.
- Then $\mathcal{I} \models q$ iff $\mathcal{I}' \models q'$.
- Query q' is acyclic and can thus be answered in polynomial time with respect to the size of the database $\mathcal{I}'$.
- But computing $\mathcal{I}'$ from $\mathcal{I}$ requires adding $|\mathbf{adom}(\mathcal{I})|^5$ new facts.
- Thus answering q' still takes exponential time with respect to the size of $\mathcal{I}$.

Exercise 4

Exercise. It is easy to see that the following is true: For a hypergraph $H = \langle V, E \rangle$, the hypergraph $H' = \langle V, E \cup \{e_V\} \rangle$ is acyclic with e_V a hyperedge that contains every vertex of V. Therefore, every BCQ can be transformed into a tree query by adding suitable atoms. Does this imply that every BCQ can be answered in polynomial time? Explain.
Solution. Let us consider a naive strategy:

- Let $q = \exists v, w, x, y, z.\ \mathsf{P}(x, y, z) \wedge \mathsf{Q}(z, w) \wedge \mathsf{R}(w, v, x) \wedge \mathsf{S}(y, w)$
- Let $q' = \exists v, w, x, y, z.\ \mathsf{P}(x, y, z) \wedge \mathsf{Q}(z, w) \wedge \mathsf{R}(w, v, x) \wedge \mathsf{S}(y, w) \wedge \mathsf{X}(v, w, x, y, z)$
- For a database instance $\mathcal{I}$, let $\mathcal{I}' = \mathcal{I} \cup \{X(c_1, \ldots, c_5) \mid c_1, \ldots, c_5 \in \textbf{adom}(\mathcal{I})\}$.
- Then $\mathcal{I} \models q$ iff $\mathcal{I}' \models q'$.
- Query q' is acyclic and can thus be answered in polynomial time with respect to the size of the database $\mathcal{I}'$.
- But computing $\mathcal{I}'$ from $\mathcal{I}$ requires adding $|\textbf{adom}(\mathcal{I})|^5$ new facts.
- Thus answering q' still takes exponential time with respect to the size of $\mathcal{I}$.
- In general, $|\textbf{adom}(\mathcal{I})|^{|V|}$ new facts are necessary.

Exercise 5.

Exercise.
Sudoku is a one-player puzzle game where one has to fill a grid with numbers. An example 4×4-Sudoku is as follows:

			3
			4
2			
3			

The grid has to be filled with numbers $\{1, 2, 3, 4\}$ such that every number occurs exactly once in each row, each column, and each 2×2-subgrid bordered in bold. For an arbitrary 4×4-Sudoku, specify a BCQ q and a database instance $\mathcal{J}$ such that $\mathcal{J} \models q$ if and only if the given Sudoku has a solution.

Exercise 5.

Exercise.
Sudoku is a one-player puzzle game where one has to fill a grid with numbers. An example 4×4-Sudoku is as follows:

			3
			4
2			
3			

The grid has to be filled with numbers $\{1, 2, 3, 4\}$ such that every number occurs exactly once in each row, each column, and each 2×2-subgrid bordered in bold. For an arbitrary 4×4-Sudoku, specify a BCQ q and a database instance $\mathcal{J}$ such that $\mathcal{J} \models q$ if and only if the given Sudoku has a solution.

Solution.

Exercise 5.

Exercise.
Sudoku is a one-player puzzle game where one has to fill a grid with numbers. An example 4×4-Sudoku is as follows:

			3
			4
2			
3			

The grid has to be filled with numbers $\{1, 2, 3, 4\}$ such that every number occurs exactly once in each row, each column, and each 2×2-subgrid bordered in bold. For an arbitrary 4×4-Sudoku, specify a BCQ q and a database instance $\mathcal{J}$ such that $\mathcal{J} \models q$ if and only if the given Sudoku has a solution.

Solution.

- We define

$$\mathcal{J} = \left\{ S(c_1^1, \ldots, c_1^4, \ldots, c_4^1, \ldots, c_4^4) \,\middle|\, c_i^j \in \begin{cases} \{\, k \,\} \text{ if position } \langle i, j \rangle \text{ contains the number } k, \text{ and} \\ \{\, 1, 2, 3, 4 \,\} \text{ otherwise} \end{cases} \right\}.$$

Exercise 5.

Exercise.
Sudoku is a one-player puzzle game where one has to fill a grid with numbers. An example 4×4-Sudoku is as follows:

			3
			4
2			
3			

The grid has to be filled with numbers $\{1, 2, 3, 4\}$ such that every number occurs exactly once in each row, each column, and each 2×2-subgrid bordered in bold. For an arbitrary 4×4-Sudoku, specify a BCQ q and a database instance $\mathcal{J}$ such that $\mathcal{J} \models q$ if and only if the given Sudoku has a solution.

Solution.

- We define

$$\mathcal{J} = \left\{ S(c_1^1, \ldots, c_1^4, \ldots, c_4^1, \ldots, c_4^4) \;\middle|\; c_i^j \in \begin{cases} \{\, k \,\} \text{ if position } \langle i, j \rangle \text{ contains the number } k, \text{ and} \\ \{\, 1, 2, 3, 4 \,\} \text{ otherwise} \end{cases} \right\}.$$

- We set $\mathbf{y} = \langle y_1^1, \ldots, y_1^4, \ldots, y_4^1, \ldots, y_4^4 \rangle$, and let q be the query

$$\exists \mathbf{y}. \Big(S(\mathbf{y}) \wedge \bigwedge_{k=1}^{4} \bigwedge_{i=1}^{4} \bigwedge_{j=i+1}^{4} \big(y_k^i \not\approx y_k^j \big) \wedge \bigwedge_{k=1}^{4} \bigwedge_{i=1}^{4} \bigwedge_{j=i+1}^{4} \big(y_i^k \not\approx y_j^k \big) \wedge \bigwedge_{i \in \{1,3\}} \bigwedge_{j \in \{1,3\}} (y_i^j \not\approx y_{i+1}^{j+1}) \wedge (y_{i+1}^j \not\approx y_i^{j+1}) \Big).$$

Exercise 5.

Exercise.
Sudoku is a one-player puzzle game where one has to fill a grid with numbers. An example 4×4-Sudoku is as follows:

			3
			4
2			
3			

The grid has to be filled with numbers $\{1, 2, 3, 4\}$ such that every number occurs exactly once in each row, each column, and each 2×2-subgrid bordered in bold. For an arbitrary 4×4-Sudoku, specify a BCQ q and a database instance $\mathcal{J}$ such that $\mathcal{J} \models q$ if and only if the given Sudoku has a solution.

Solution.

- We define

$$\mathcal{J} = \left\{ S(c_1^1, \ldots, c_1^4, \ldots, c_4^1, \ldots, c_4^4) \;\middle|\; c_i^j \in \begin{cases} \{\, k \,\} \text{ if position } \langle i, j \rangle \text{ contains the number } k, \text{ and} \\ \{\, 1, 2, 3, 4 \,\} \text{ otherwise} \end{cases} \right\}.$$

- We set $\mathbf{y} = \langle y_1^1, \ldots, y_1^4, \ldots, y_4^1, \ldots, y_4^4 \rangle$, and let q be the query

$$\exists \mathbf{y}. \Big(S(\mathbf{y}) \wedge \bigwedge_{k=1}^{4} \bigwedge_{i=1}^{4} \bigwedge_{j=i+1}^{4} \big(y_k^i \not\approx y_k^j\big) \wedge \bigwedge_{k=1}^{4} \bigwedge_{i=1}^{4} \bigwedge_{j=i+1}^{4} \big(y_i^k \not\approx y_j^k\big) \wedge \bigwedge_{i \in \{1,3\}} \bigwedge_{j \in \{1,3\}} (y_i^j \not\approx y_{i+1}^{j+1}) \wedge (y_{i+1}^j \not\approx y_i^{j+1}) \Big).$$

- Then $\mathcal{J} \models q$ iff the Sudoku has a solution.

Exercise 6.

Exercise. It was shown in the lecture that the 3-colourability problem for graphs can be reduced to the homomorphism problem. Therefore, it can also be expressed as a BCQ answering problem.

1. In which cases is the resulting BCQ a tree query?
2. What is the complexity of solving the 3-colourability problem for these cases?

Exercise 6.

Exercise. It was shown in the lecture that the 3-colourability problem for graphs can be reduced to the homomorphism problem. Therefore, it can also be expressed as a BCQ answering problem.

1. In which cases is the resulting BCQ a tree query?
2. What is the complexity of solving the 3-colourability problem for these cases?

Reduction.

Exercise 6.

Exercise. It was shown in the lecture that the 3-colourability problem for graphs can be reduced to the homomorphism problem. Therefore, it can also be expressed as a BCQ answering problem.

1. In which cases is the resulting BCQ a tree query?
2. What is the complexity of solving the 3-colourability problem for these cases?

Reduction. Recall the definition of the two decision problems:

$$\text{3C} = \{\langle G\rangle \mid G \text{ is three-colourable}\} \qquad \text{QE} = \{\langle \mathcal{I}, q\rangle \mid \mathcal{I} \models q\}$$

Exercise 6.

Exercise. It was shown in the lecture that the 3-colourability problem for graphs can be reduced to the homomorphism problem. Therefore, it can also be expressed as a BCQ answering problem.

1. In which cases is the resulting BCQ a tree query?
2. What is the complexity of solving the 3-colourability problem for these cases?

Reduction. Recall the definition of the two decision problems:

$$3C = \{ \langle G \rangle \mid G \text{ is three-colourable} \} \qquad QE = \{ \langle \mathcal{I}, q \rangle \mid \mathcal{I} \models q \}$$

- Given a graph $G = \langle V, E \rangle$, let $\mathcal{I} = \{E(r, b), E(b, g), E(g, r)\}$ and let q be the BCQ $\exists_{v \in V} x_v . \bigwedge_{\langle v,u \rangle \in E} E(x_v, x_u)$ containing the atom $E(x_v, x_u)$ if and only if there is an edge between v and u in G.

Exercise 6.

Exercise. It was shown in the lecture that the 3-colourability problem for graphs can be reduced to the homomorphism problem. Therefore, it can also be expressed as a BCQ answering problem.

1. In which cases is the resulting BCQ a tree query?
2. What is the complexity of solving the 3-colourability problem for these cases?

Reduction. Recall the definition of the two decision problems:

$$3C = \{ \langle G \rangle \mid G \text{ is three-colourable} \} \qquad QE = \{ \langle \mathcal{I}, q \rangle \mid \mathcal{I} \models q \}$$

- Given a graph $G = \langle V, E \rangle$, let $\mathcal{I} = \{E(r,b), E(b,g), E(g,r)\}$ and let q be the BCQ $\exists_{v \in V} x_v . \bigwedge_{\langle v,u \rangle \in E} E(x_v, x_u)$ containing the atom $E(x_v, x_u)$ if and only if there is an edge between v and u in G.
- Then $\langle G \rangle \in 3C$ iff $\langle \mathcal{I}, q \rangle \in QE$.

Exercise 6.

Exercise. It was shown in the lecture that the 3-colourability problem for graphs can be reduced to the homomorphism problem. Therefore, it can also be expressed as a BCQ answering problem.

1. In which cases is the resulting BCQ a tree query?
2. What is the complexity of solving the 3-colourability problem for these cases?

Reduction. Recall the definition of the two decision problems:

$$3C = \{ \langle G \rangle \mid G \text{ is three-colourable} \} \qquad QE = \{ \langle \mathcal{I}, q \rangle \mid \mathcal{I} \models q \}$$

- Given a graph $G = \langle V, E \rangle$, let $\mathcal{I} = \{E(r, b), E(b, g), E(g, r)\}$ and let q be the BCQ $\exists_{v \in V} x_v . \bigwedge_{\langle v,u \rangle \in E} E(x_v, x_u)$ containing the atom $E(x_v, x_u)$ if and only if there is an edge between v and u in G.
- Then $\langle G \rangle \in 3C$ iff $\langle \mathcal{I}, q \rangle \in QE$.

Solution.

Exercise 6.

Exercise. It was shown in the lecture that the 3-colourability problem for graphs can be reduced to the homomorphism problem. Therefore, it can also be expressed as a BCQ answering problem.

1. In which cases is the resulting BCQ a tree query?
2. What is the complexity of solving the 3-colourability problem for these cases?

Reduction. Recall the definition of the two decision problems:

$$3\text{C} = \{ \langle G \rangle \mid G \text{ is three-colourable} \} \qquad \text{QE} = \{ \langle \mathcal{I}, q \rangle \mid \mathcal{I} \models q \}$$

- Given a graph $G = \langle V, E \rangle$, let $\mathcal{I} = \{E(r, b), E(b, g), E(g, r)\}$ and let q be the BCQ $\exists_{v \in V} x_v . \bigwedge_{\langle v,u \rangle \in E} E(x_v, x_u)$ containing the atom $E(x_v, x_u)$ if and only if there is an edge between v and u in G.
- Then $\langle G \rangle \in 3\text{C}$ iff $\langle \mathcal{I}, q \rangle \in \text{QE}$.

Solution.

1. q is a tree query when G is acyclic.

Exercise 6.

Exercise. It was shown in the lecture that the 3-colourability problem for graphs can be reduced to the homomorphism problem. Therefore, it can also be expressed as a BCQ answering problem.

1. In which cases is the resulting BCQ a tree query?
2. What is the complexity of solving the 3-colourability problem for these cases?

Reduction. Recall the definition of the two decision problems:

$$3\text{C} = \{\langle G\rangle \mid G \text{ is three-colourable}\} \qquad \text{QE} = \{\langle \mathcal{I}, q\rangle \mid \mathcal{I} \models q\}$$

- Given a graph $G = \langle V, E\rangle$, let $\mathcal{I} = \{E(r,b), E(b,g), E(g,r)\}$ and let q be the BCQ $\exists_{v\in V} x_v. \bigwedge_{\langle v,u\rangle\in E} E(x_v, x_u)$ containing the atom $E(x_v, x_u)$ if and only if there is an edge between v and u in G.
- Then $\langle G\rangle \in 3\text{C}$ iff $\langle \mathcal{I}, q\rangle \in \text{QE}$.

Solution.

1. q is a tree query when G is acyclic.
2. If G is acyclic, then q can be answered in constant time.

Exercise 7.

Exercise. A propositional formula is in **3CNF** if it has the following form:

$$(L_1^1 \vee L_2^1 \vee L_3^1) \wedge (L_1^2 \vee L_2^2 \vee L_3^2) \wedge \cdots \wedge (L_1^n \vee L_2^n \vee L_3^n),$$

where each L is a literal, that is, a propositional variable or the negation of a propositional variable. The **3SAT** problem is the problem of deciding if a given **3CNF** formula is satisfiable. It is known to be NP-complete.
Reduce **3SAT** to the homomorphism problem: define a suitable hypergraph $\mathcal{I}_\varphi$ for every **3CNF** φ and give a template $\mathcal{J}$, such that there is a homomorphism from $\mathcal{I}_\varphi$ to $\mathcal{J}$ iff φ is satisfiable (the template $\mathcal{J}$ can be the same for all inputs.)

Exercise 7.

Exercise. A propositional formula is in **3CNF** if it has the following form:

$$(L_1^1 \vee L_2^1 \vee L_3^1) \wedge (L_1^2 \vee L_2^2 \vee L_3^2) \wedge \cdots \wedge (L_1^n \vee L_2^n \vee L_3^n),$$

where each L is a literal, that is, a propositional variable or the negation of a propositional variable. The **3SAT** problem is the problem of deciding if a given **3CNF** formula is satisfiable. It is known to be NP-complete.
Reduce **3SAT** to the homomorphism problem: define a suitable hypergraph $\mathcal{I}_\varphi$ for every **3CNF** φ and give a template $\mathcal{J}$, such that there is a homomorphism from $\mathcal{I}_\varphi$ to $\mathcal{J}$ iff φ is satisfiable (the template $\mathcal{J}$ can be the same for all inputs.)
Remark. This yields an alternative proof of the NP-hardness of the homomorphism problem.

Exercise 7.

Exercise. A propositional formula is in **3CNF** if it has the following form:

$$(L_1^1 \vee L_2^1 \vee L_3^1) \wedge (L_1^2 \vee L_2^2 \vee L_3^2) \wedge \cdots \wedge (L_1^n \vee L_2^n \vee L_3^n),$$

where each L is a literal, that is, a propositional variable or the negation of a propositional variable. The **3SAT** problem is the problem of deciding if a given **3CNF** formula is satisfiable. It is known to be NP-complete.
Reduce **3SAT** to the homomorphism problem: define a suitable hypergraph $\mathcal{I}_\varphi$ for every **3CNF** φ and give a template $\mathcal{J}$, such that there is a homomorphism from $\mathcal{I}_\varphi$ to $\mathcal{J}$ iff φ is satisfiable (the template $\mathcal{J}$ can be the same for all inputs.)
Remark. This yields an alternative proof of the NP-hardness of the homomorphism problem.
Solution.

Exercise 7.

Exercise. A propositional formula is in **3CNF** if it has the following form:

$$(L_1^1 \vee L_2^1 \vee L_3^1) \wedge (L_1^2 \vee L_2^2 \vee L_3^2) \wedge \cdots \wedge (L_1^n \vee L_2^n \vee L_3^n),$$

where each L is a literal, that is, a propositional variable or the negation of a propositional variable. The **3SAT** problem is the problem of deciding if a given **3CNF** formula is satisfiable. It is known to be NP-complete.
Reduce **3SAT** to the homomorphism problem: define a suitable hypergraph $\mathcal{I}_\varphi$ for every **3CNF** φ and give a template $\mathcal{J}$, such that there is a homomorphism from $\mathcal{I}_\varphi$ to $\mathcal{J}$ iff φ is satisfiable (the template $\mathcal{J}$ can be the same for all inputs.)
Remark. This yields an alternative proof of the NP-hardness of the homomorphism problem.
Solution.

- $\mathcal{I}_\varphi$ is the hypergraph consisting of all edges $C(L_1^i, L_2^i, L_3^i)$ with $1 \leq i \leq n$ and all edges $V(L, \neg L)$ for every positive literal L syntactically occurring in φ.

Exercise 7.

Exercise. A propositional formula is in **3CNF** if it has the following form:

$$(L_1^1 \vee L_2^1 \vee L_3^1) \wedge (L_1^2 \vee L_2^2 \vee L_3^2) \wedge \cdots \wedge (L_1^n \vee L_2^n \vee L_3^n),$$

where each L is a literal, that is, a propositional variable or the negation of a propositional variable. The **3SAT** problem is the problem of deciding if a given **3CNF** formula is satisfiable. It is known to be NP-complete.
Reduce **3SAT** to the homomorphism problem: define a suitable hypergraph $\mathcal{I}_\varphi$ for every **3CNF** φ and give a template $\mathcal{J}$, such that there is a homomorphism from $\mathcal{I}_\varphi$ to $\mathcal{J}$ iff φ is satisfiable (the template $\mathcal{J}$ can be the same for all inputs.)
Remark. This yields an alternative proof of the NP-hardness of the homomorphism problem.
Solution.

- $\mathcal{I}_\varphi$ is the hypergraph consisting of all edges $C(L_1^i, L_2^i, L_3^i)$ with $1 \leq i \leq n$ and all edges $V(L, \neg L)$ for every positive literal L syntactically occurring in φ.
- $\mathcal{J} = \left\{ C(x, y, z) \,\middle|\, x, y, z \in \{0, 1\} \text{ with } x + y + z > 0 \right\} \cup \left\{ V(0, 1), V(1, 0) \right\}$

Exercise 7.

Exercise. A propositional formula is in **3CNF** if it has the following form:

$$(L_1^1 \vee L_2^1 \vee L_3^1) \wedge (L_1^2 \vee L_2^2 \vee L_3^2) \wedge \cdots \wedge (L_1^n \vee L_2^n \vee L_3^n),$$

where each L is a literal, that is, a propositional variable or the negation of a propositional variable. The **3SAT** problem is the problem of deciding if a given **3CNF** formula is satisfiable. It is known to be NP-complete.
Reduce **3SAT** to the homomorphism problem: define a suitable hypergraph $\mathcal{I}_\varphi$ for every **3CNF** φ and give a template $\mathcal{J}$, such that there is a homomorphism from $\mathcal{I}_\varphi$ to $\mathcal{J}$ iff φ is satisfiable (the template $\mathcal{J}$ can be the same for all inputs.)
Remark. This yields an alternative proof of the NP-hardness of the homomorphism problem.
Solution.

- $\mathcal{I}_\varphi$ is the hypergraph consisting of all edges $C(L_1^i, L_2^i, L_3^i)$ with $1 \leq i \leq n$ and all edges $V(L, \neg L)$ for every positive literal L syntactically occurring in φ.
- $\mathcal{J} = \left\{ C(x, y, z) \,\middle|\, x, y, z \in \{0, 1\} \text{ with } x + y + z > 0 \right\} \cup \left\{ V(0, 1), V(1, 0) \right\}$
- Then φ is satisfiable iff there is a homomorphsim from $\mathcal{I}_\varphi to \mathcal{J}$.