Model-checking hierarchical graphs

Markus Lohrey
University of Stuttgart

28. Juli 2005

Hierarchical graph definitions (HGDs) I

Hierarchical graph definitions (HGDs) I

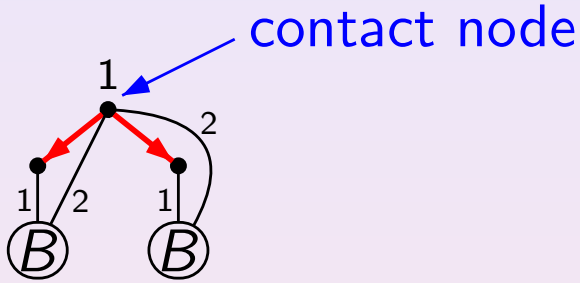
An HGD D

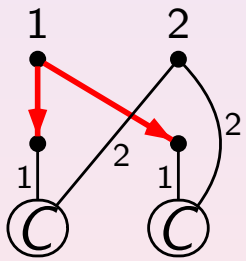
$S :=$ 

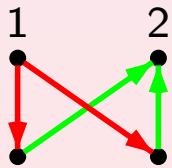
Macros: $S \succ A \succ B \succ C$

$\text{rank}(S)=0$, $\text{rank}(A)=1$

$\text{rank}(B) = \text{rank}(C) = 2$

$A :=$ 

$B :=$ 

$C :=$ 

Hierarchical graph definitions (HGDs) I

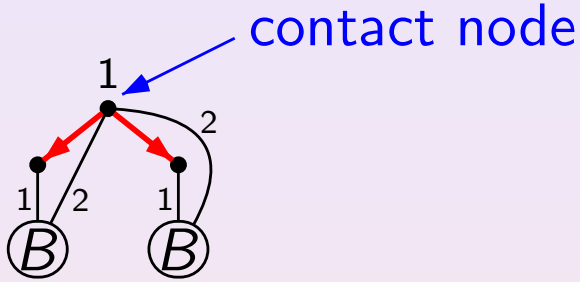
An HGD D

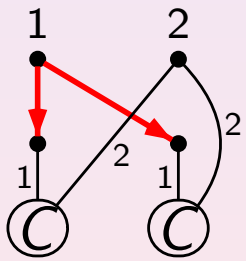
$S :=$ 

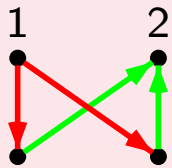
Macros: $S \succ A \succ B \succ C$

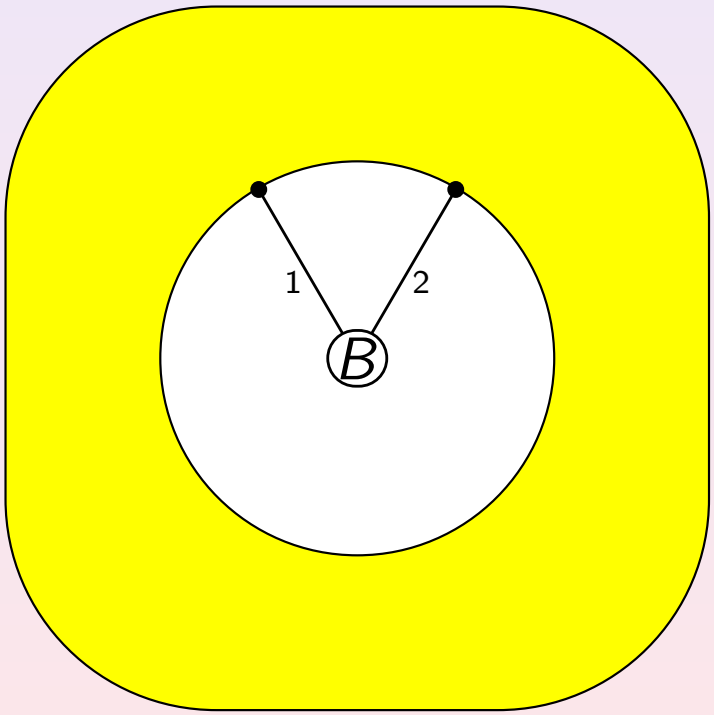
rank(S)=0, rank(A)=1

rank(B) = rank(C) = 2

$A :=$ 

$B :=$ 

$C :=$ 



Hierarchical graph definitions (HGDs) I

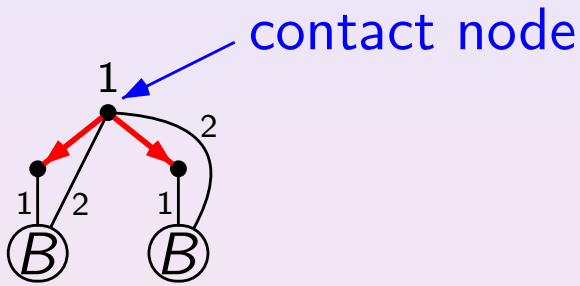
An HGD D

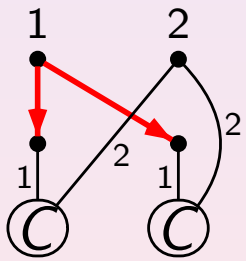
$S :=$ 

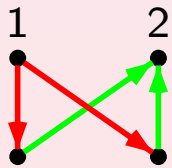
Macros: $S \succ A \succ B \succ C$

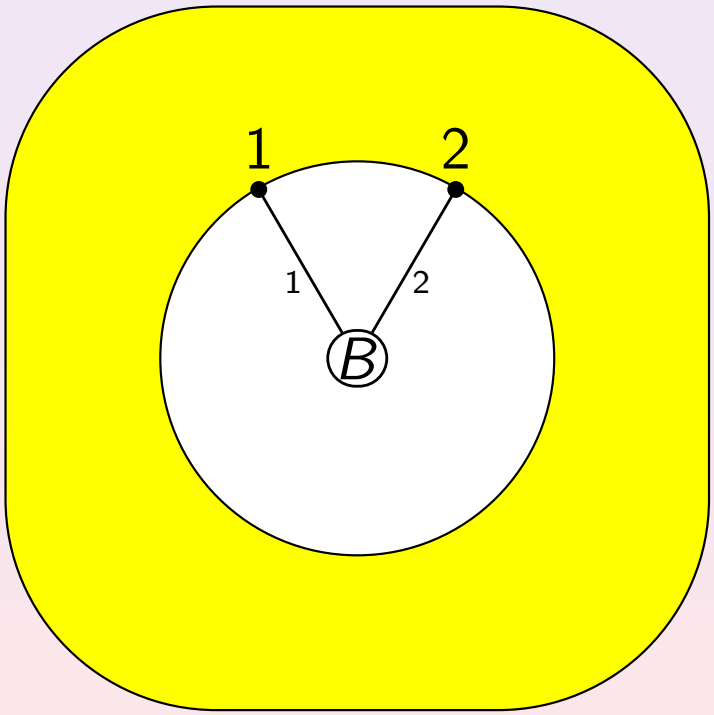
rank(S)=0, rank(A)=1

rank(B) = rank(C) = 2

$A :=$ 

$B :=$ 

$C :=$ 



Hierarchical graph definitions (HGDs) I

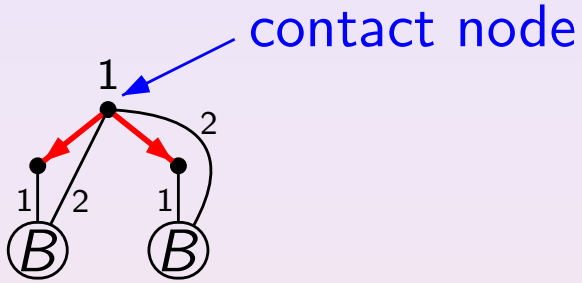
An HGD D

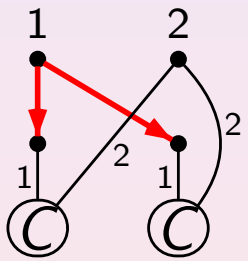
$S :=$ 

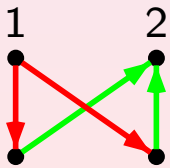
Macros: $S \succ A \succ B \succ C$

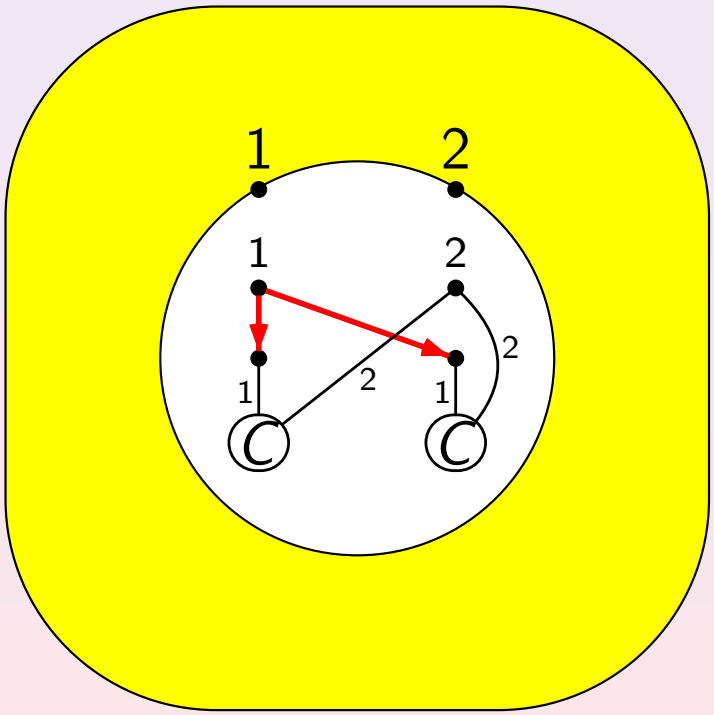
rank(S)=0, rank(A)=1

rank(B) = rank(C) = 2

$A :=$ 

$B :=$ 

$C :=$ 



Hierarchical graph definitions (HGDs) I

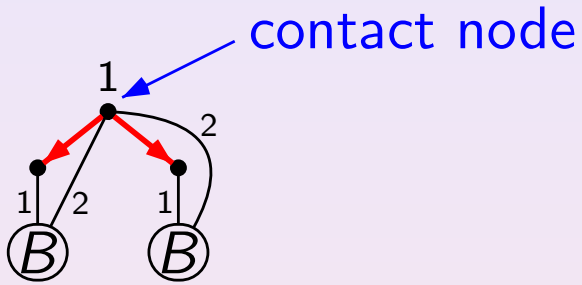
An HGD D

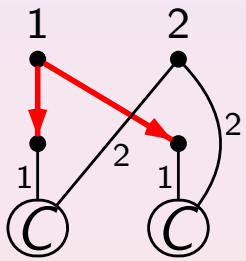
$S :=$ 

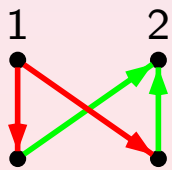
Macros: $S \succ A \succ B \succ C$

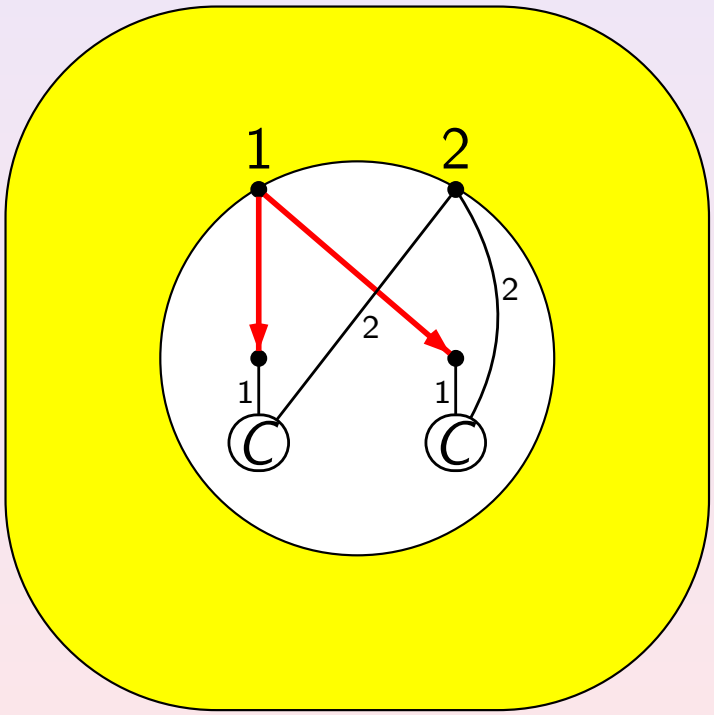
$\text{rank}(S)=0$, $\text{rank}(A)=1$

$\text{rank}(B) = \text{rank}(C) = 2$

$A :=$ 

$B :=$ 

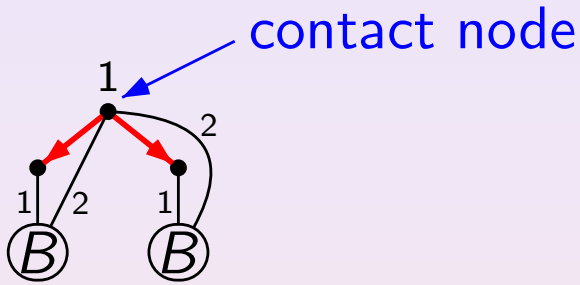
$C :=$ 

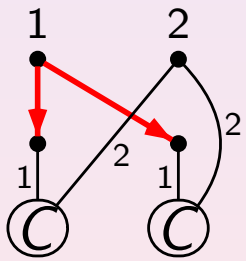


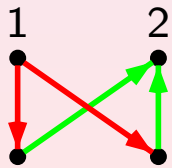
Hierarchical graph definitions (HGDs) I

An HGD D

$S :=$ 

$A :=$ 

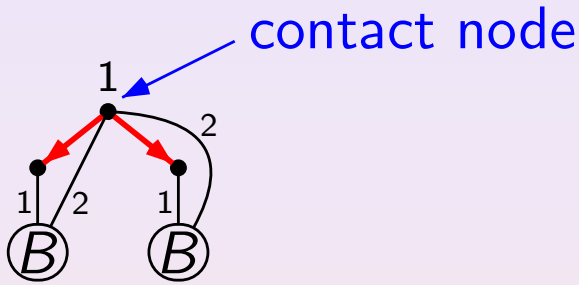
$B :=$ 

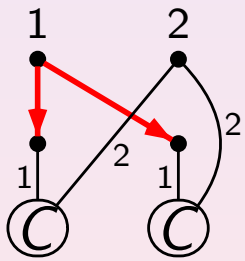
$C :=$ 

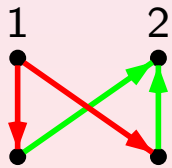
Hierarchical graph definitions (HGDs) I

An HGD D

$S :=$ 

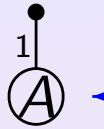
$A :=$ 

$B :=$ 

$C :=$ 

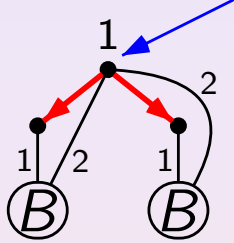
Hierarchical graph definitions (HGDs) I

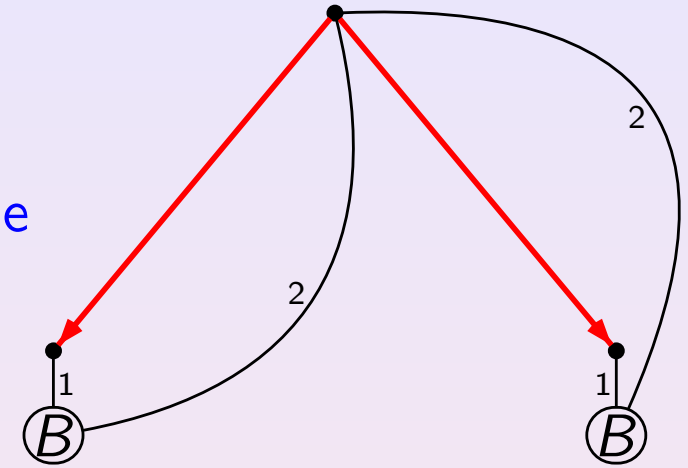
An HGD D

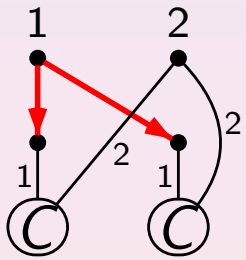
$S :=$ 

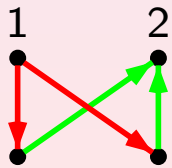
macro

contact node

$A :=$ 

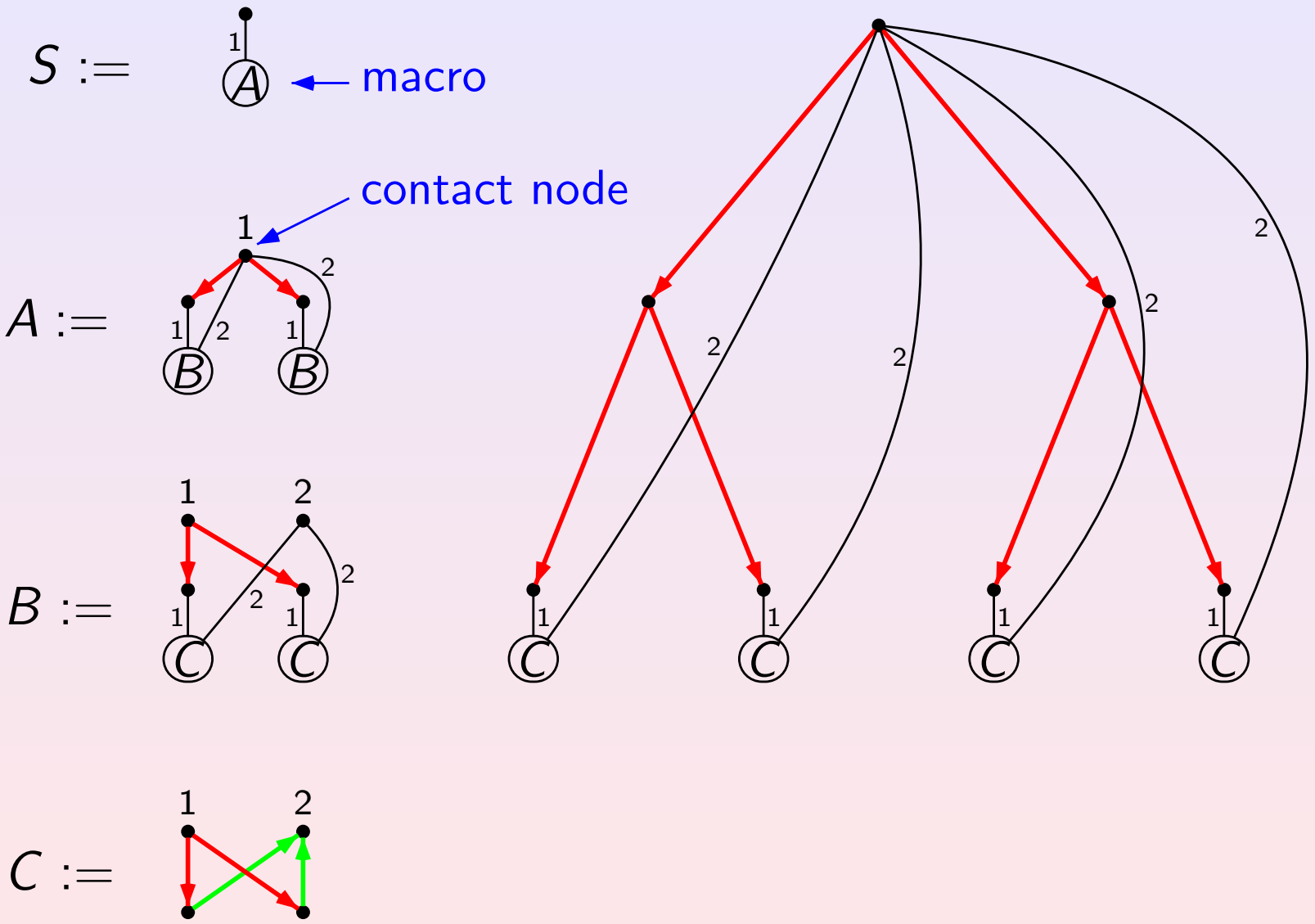


$B :=$ 

$C :=$ 

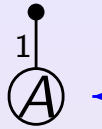
Hierarchical graph definitions (HGDs) I

An HGD D



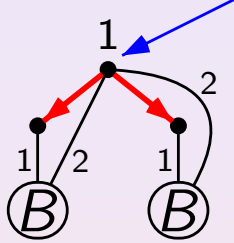
Hierarchical graph definitions (HGDs) I

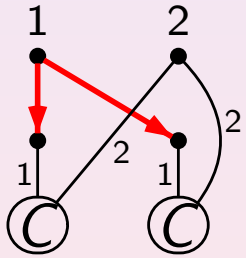
An HGD D

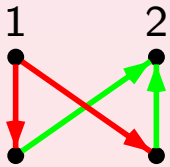
$S :=$ 

macro

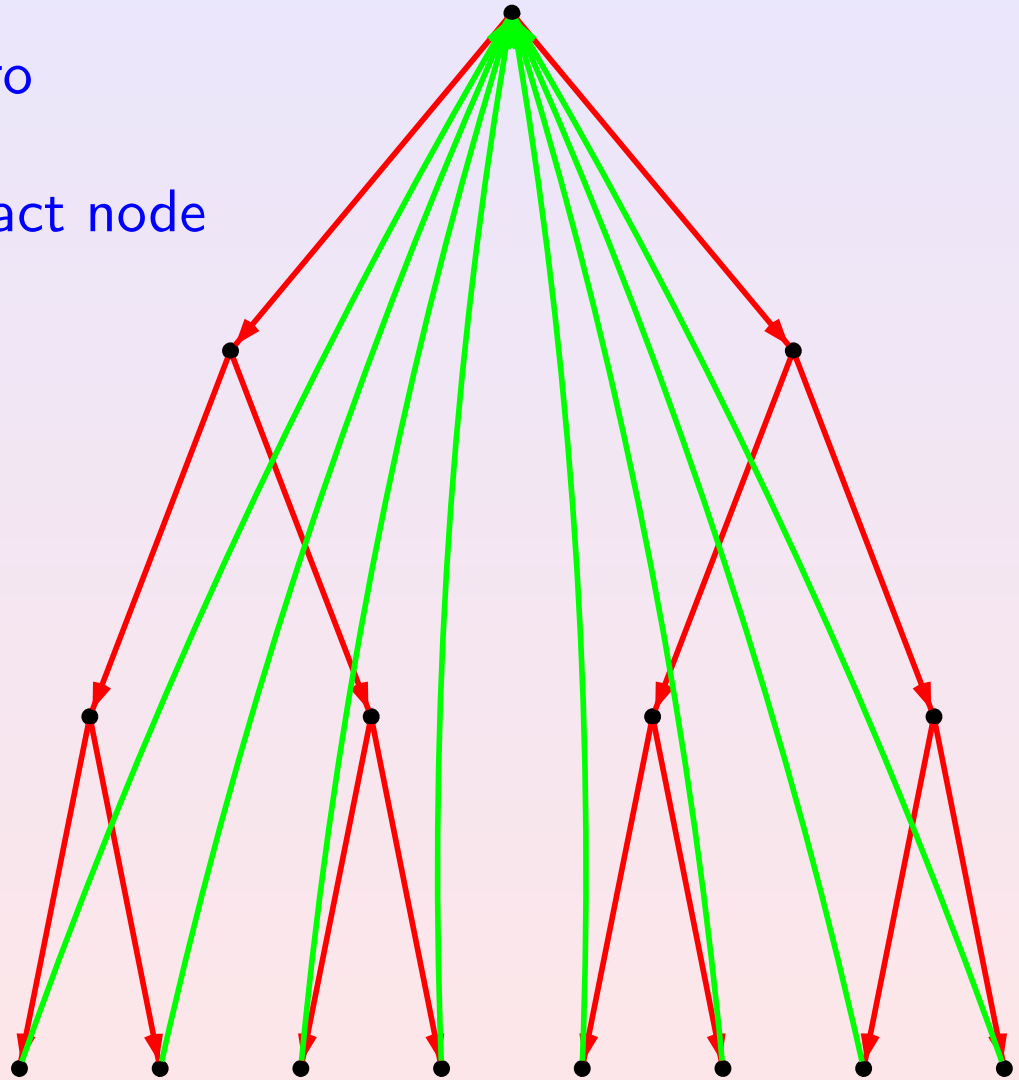
contact node

$A :=$ 

$B :=$ 

$C :=$ 

unfold(D)



Hierarchical graph definitions (HGDs) II

For an HGD D, $\mathrm{unfold}(D)$ denotes the generated graph.

Hierarchical graph definitions (HGDs) II

For an HGD D, $\mathrm{unfold}(D)$ denotes the generated graph.

Data compression (succinctness): $\mathrm{unfold}(D)$ may be exponentially larger than D.

Hierarchical graph definitions (HGDs) II

For an HGD D, $\mathrm{unfold}(D)$ denotes the generated graph.

Data compression (succinctness): $\mathrm{unfold}(D)$ may be exponentially larger than D.

Consequence: Computational problems tend to be more difficult, when input structures are given hierarchical!

Hierarchical graph definitions (HGDs) II

For an HGD D, $\mathrm{unfold}(D)$ denotes the generated graph.

Data compression (succinctness): $\mathrm{unfold}(D)$ may be exponentially larger than D.

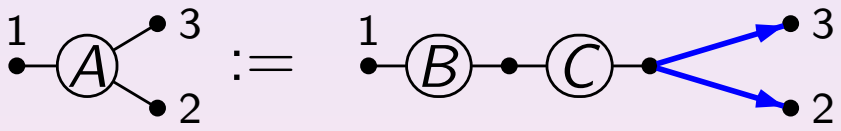
Consequence: Computational problems tend to be more difficult, when input structures are given hierarchical!

W.l.o.g. at most two references in each right-hand side (Chomsky normal form)

Hierarchical graph definitions (HGDs) III

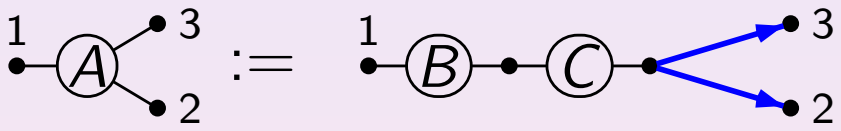
Hierarchical graph definitions (HGDs) III

A HGD D is non-passing if macros are not connected to contact nodes in right-hand sides.

E.g. not allowed: 

Hierarchical graph definitions (HGDs) III

A HGD D is non-passing if macros are not connected to contact nodes in right-hand sides.

E.g. not allowed: 

A HGD D is c-bounded for $c \in \mathbb{N}$ if the rank of every macro is at most c.

History and related formalisms

Previous work:

- Lengauer, Wanke (around 1990): Specific algorithmic problems on hierarchically defined graphs.
- Marathe, Hunt, Stearns, Radhakrishnan (around 1995): Approximation algorithms for hierarchically defined graphs.
- Alur, Yannakakis 2001: Model-checking temporal logics for hierarchical state machines.
- Genest, Muscholl, L: Hierarchical message sequence charts

History and related formalisms

Previous work:

- Lengauer, Wanke (around 1990): Specific algorithmic problems on hierarchically defined graphs.
- Marathe, Hunt, Stearns, Radhakrishnan (around 1995): Approximation algorithms for hierarchically defined graphs.
- Alur, Yannakakis 2001: Model-checking temporal logics for hierarchical state machines.
- Genest, Muscholl, L: Hierarchical message sequence charts

Here: Model-Checking of first-order logic (FO) and monadic second-order logic (MSO) on hierarchically defined input graphs.

FO and MSO

First-order logic (FO): quantification over elements of a structure

FO and MSO

First-order logic (FO): quantification over elements of a structure
Example ((V, E) is a directed graph):

$$\forall x \in V \, \exists y_1 \in V \, \exists y_2 \in V : y_1 \neq y_2 \ \wedge \ E(x, y_1) \ \wedge \ E(x, y_2) \wedge$$
$$\forall z \in V : E(x, z) \Rightarrow z = y_1 \vee z = y_2$$

FO and MSO

First-order logic (FO): quantification over elements of a structure
Example ((V, E) is a directed graph):

$$\forall x \in V \, \exists y_1 \in V \, \exists y_2 \in V : y_1 \neq y_2 \ \wedge \ E(x, y_1) \ \wedge \ E(x, y_2) \wedge \forall z \in V : E(x, z) \Rightarrow z = y_1 \vee z = y_2$$

Monadic second-order logic (MSO): quantification over elements and subsets of a structure

FO and MSO

First-order logic (FO): quantification over elements of a structure
Example ((V, E) is a directed graph):

$$\forall x \in V \, \exists y_1 \in V \, \exists y_2 \in V : y_1 \neq y_2 \ \wedge \ E(x, y_1) \ \wedge \ E(x, y_2) \wedge \\ \forall z \in V : E(x, z) \Rightarrow z = y_1 \vee z = y_2$$

Monadic second-order logic (MSO): quantification over elements and subsets of a structure
Example:

$$\exists x \in V \, \exists y \in V \, \exists X \subseteq V : x \in X \wedge y \notin Y \wedge \\ \forall u \in V \, \forall v \in V : u \in X \wedge E(u, v) \Rightarrow v \in X$$

Data complexity versus combined complexity

For the model-checking problem of logic $\mathcal{L}$ we distinguish:

Data complexity versus combined complexity

For the model-checking problem of logic $\mathcal{L}$ we distinguish:

- Combined complexity:
  INPUT: Formula $\varphi \in \mathcal{L}$, finite structure $\mathcal{A}$
  QUESTION: $\mathcal{A} \models \varphi$?

Data complexity versus combined complexity

For the model-checking problem of logic $\mathcal{L}$ we distinguish:

- Combined complexity:
  INPUT: Formula $\varphi \in \mathcal{L}$, finite structure $\mathcal{A}$
  QUESTION: $\mathcal{A} \models \varphi$?
- Data complexity: Formula $\varphi \in \mathcal{L}$ is fixed:
  INPUT: Finite structure $\mathcal{A}$
  QUESTION: $\mathcal{A} \models \varphi$?

Data complexity versus combined complexity

For the model-checking problem of logic $\mathcal{L}$ we distinguish:

- Combined complexity:
  INPUT: Formula $\varphi \in \mathcal{L}$, finite structure $\mathcal{A}$
  QUESTION: $\mathcal{A} \models \varphi$?
- Data complexity: Formula $\varphi \in \mathcal{L}$ is fixed:
  INPUT: Finite structure $\mathcal{A}$
  QUESTION: $\mathcal{A} \models \varphi$?

A Σ_k-FO formula is an FO-formula of the form

$$\underbrace{\exists^*\forall^*\exists^* \cdots (\exists/\forall)^*}_{k \text{ blocks of FO-quantifiers}} : \underbrace{\psi}_{\text{quantifier-free}}$$

Data complexity versus combined complexity

For the model-checking problem of logic $\mathcal{L}$ we distinguish:

- Combined complexity:
  INPUT: Formula $\varphi \in \mathcal{L}$, finite structure $\mathcal{A}$
  QUESTION: $\mathcal{A} \models \varphi$?
- Data complexity: Formula $\varphi \in \mathcal{L}$ is fixed:
  INPUT: Finite structure $\mathcal{A}$
  QUESTION: $\mathcal{A} \models \varphi$?

A Σ_k-FO formula is an FO-formula of the form

$$\underbrace{\exists^*\forall^*\exists^* \cdots (\exists/\forall)^*}_{k \text{ blocks of FO-quantifiers}} : \underbrace{\psi}_{\text{quantifier-free}}$$

A Σ_k-MSO formula is an MSO-formula of the form

$$\underbrace{\exists^*\forall^*\exists^* \cdots (\exists/\forall)^*}_{k \text{ blocks of MSO-quantifiers}} : \underbrace{\psi}_{\text{only FO-quantifiers}}$$

Some complexity classes

Some complexity classes

NL: nondeterministic logspace

Some complexity classes

NL: nondeterministic logspace

P: deterministic polynomial time

Some complexity classes

NL: nondeterministic logspace

P: deterministic polynomial time

Σ_k^{poly}: the k-th level of the polynomial time hierarchy

Some complexity classes

NL: nondeterministic logspace

P: deterministic polynomial time

Σ_k^{poly}: the k-th level of the polynomial time hierarchy
= the class of all problems that can be accepted by an alternating Turing machine in

- polynomial time and
- $k-1$ alternations starting in an existential state

Some complexity classes

NL: nondeterministic logspace

P: deterministic polynomial time

Σ_k^{poly}: the k-th level of the polynomial time hierarchy
= the class of all problems that can be accepted by an alternating Turing machine in

- polynomial time and
- $k-1$ alternations starting in an existential state

Recall: NP $= \Sigma_1^{\text{poly}}$

Some complexity classes

NL: nondeterministic logspace

P: deterministic polynomial time

Σ_k^{poly}: the k-th level of the polynomial time hierarchy
= the class of all problems that can be accepted by an alternating Turing machine in

- polynomial time and
- $k-1$ alternations starting in an existential state

Recall: NP $= \Sigma_1^{\text{poly}}$

Σ_k^{exp}: the k-th level of the exponential time hierarchy

Some complexity classes

NL: nondeterministic logspace

P: deterministic polynomial time

Σ_k^{poly}: the k-th level of the polynomial time hierarchy
= the class of all problems that can be accepted by an alternating Turing machine in

- polynomial time and
- $k-1$ alternations starting in an existential state

Recall: NP $= \Sigma_1^{\text{poly}}$

Σ_k^{exp}: the k-th level of the exponential time hierarchy
= the class of all problems that can be accepted by an alternating Turing machine in

- exponential time and
- $k-1$ alternations starting in an existential state

Complexity of FO

data complexity	non-hierarchical	apex HGD	c-bounded HGD	general HGD
Σ_k-FO	AC^0_k	NL	NL $\cdots$ P	NL $(k = 1)$ $\Sigma^{\mathrm{poly}}_{k-1}$ $(k > 1)$

Complexity of FO

data complexity	non-hierarchical	apex HGD	c-bounded HGD	general HGD
Σ_k-FO	AC^0_k	NL	NL $\cdots$ P	NL $(k = 1)$ $\Sigma^{\mathrm{poly}}_{k-1}$ $(k > 1)$

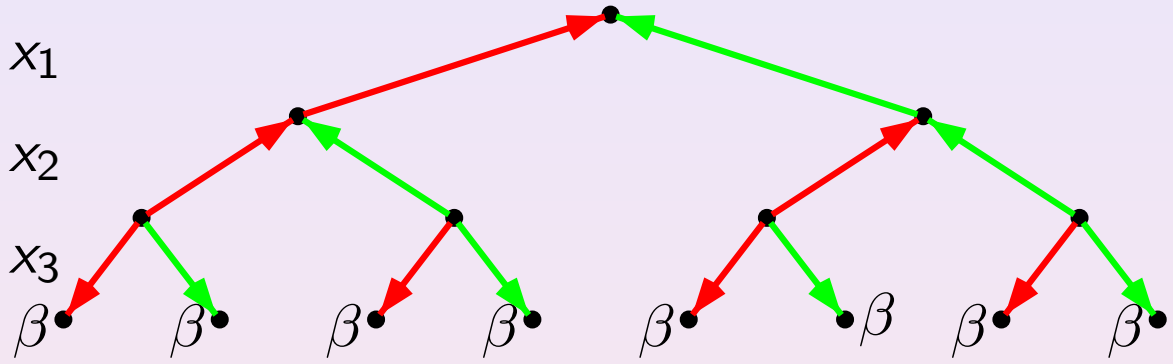
combined complexity	non-hierarchical	apex HGD	c-bounded HGD	general HGD
Σ_k-FO	Σ^{poly}_k			

Data complexity of Σ_2-FO is NP-hard

Data complexity of Σ_2-FO is NP-hard

Reduction from the NP-hard problem 3-SAT

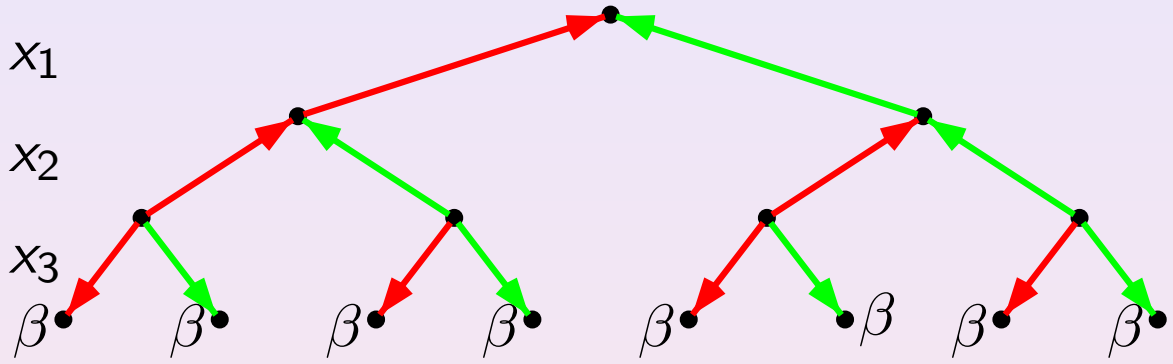
Example: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$



Data complexity of Σ_2-FO is NP-hard

Reduction from the NP-hard problem 3-SAT

Example: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

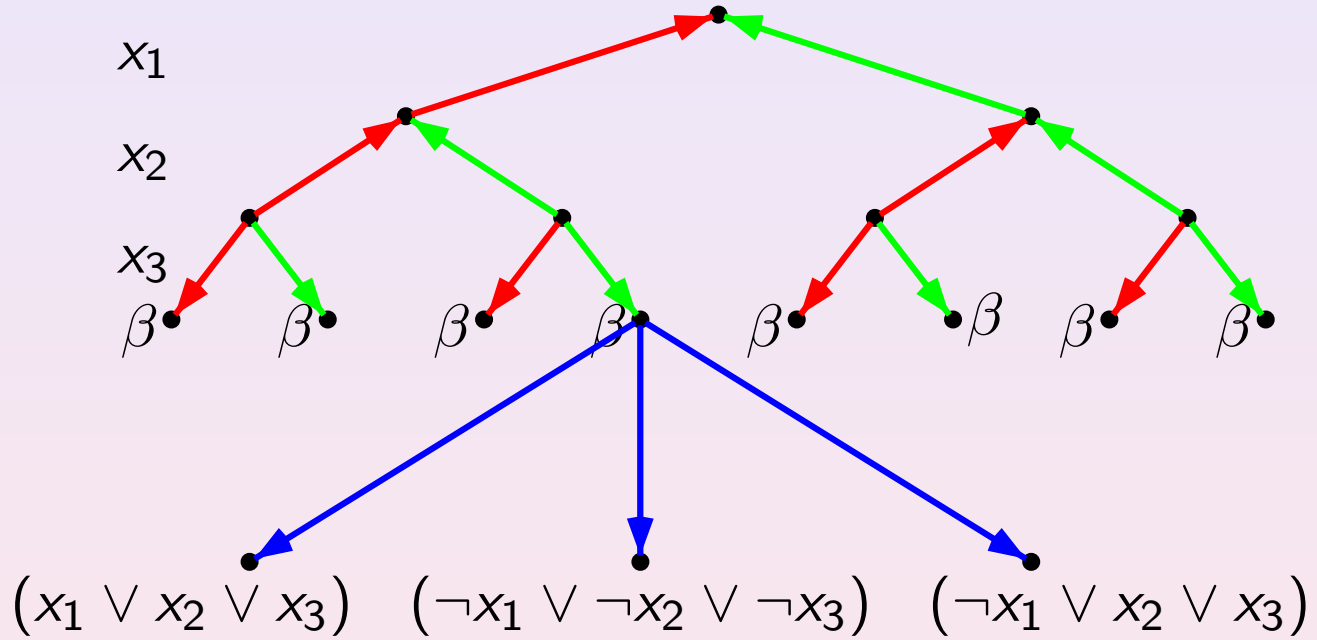


$\exists x : \beta(x) \wedge \cdots$

Data complexity of Σ_2-FO is NP-hard

Reduction from the NP-hard problem 3-SAT

Example: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

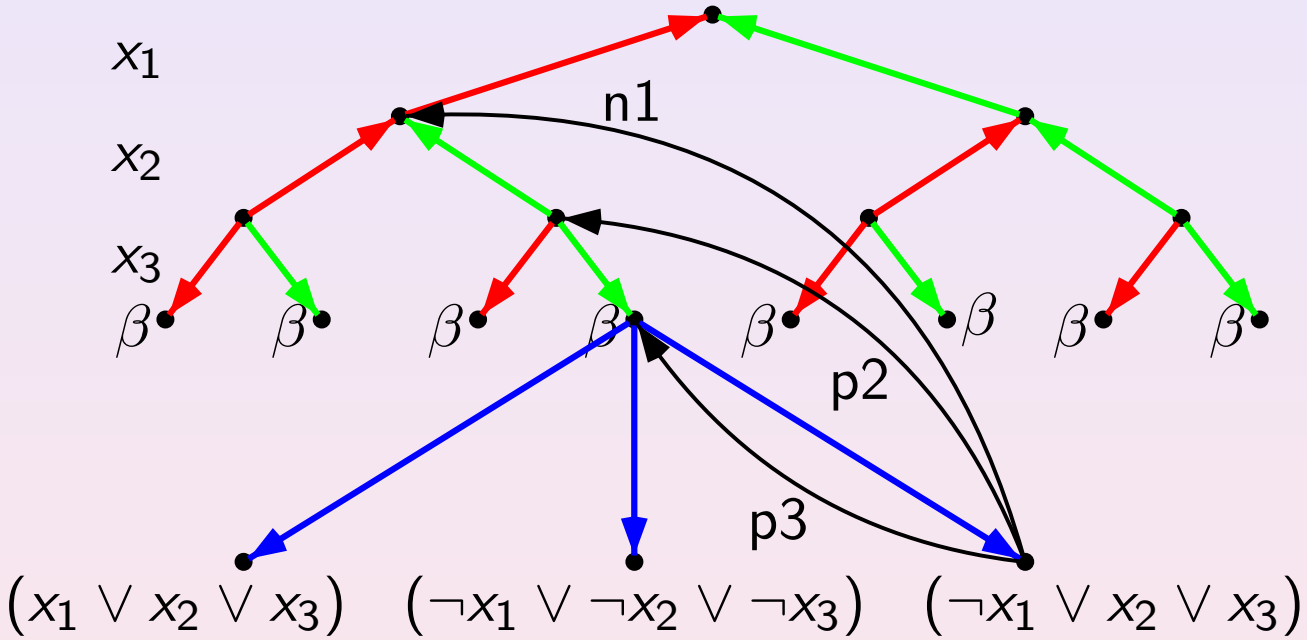


$\exists x : \beta(x) \wedge \forall y : x \longrightarrow y \Rightarrow \cdots$

Data complexity of Σ_2-FO is NP-hard

Reduction from the NP-hard problem 3-SAT

Example: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

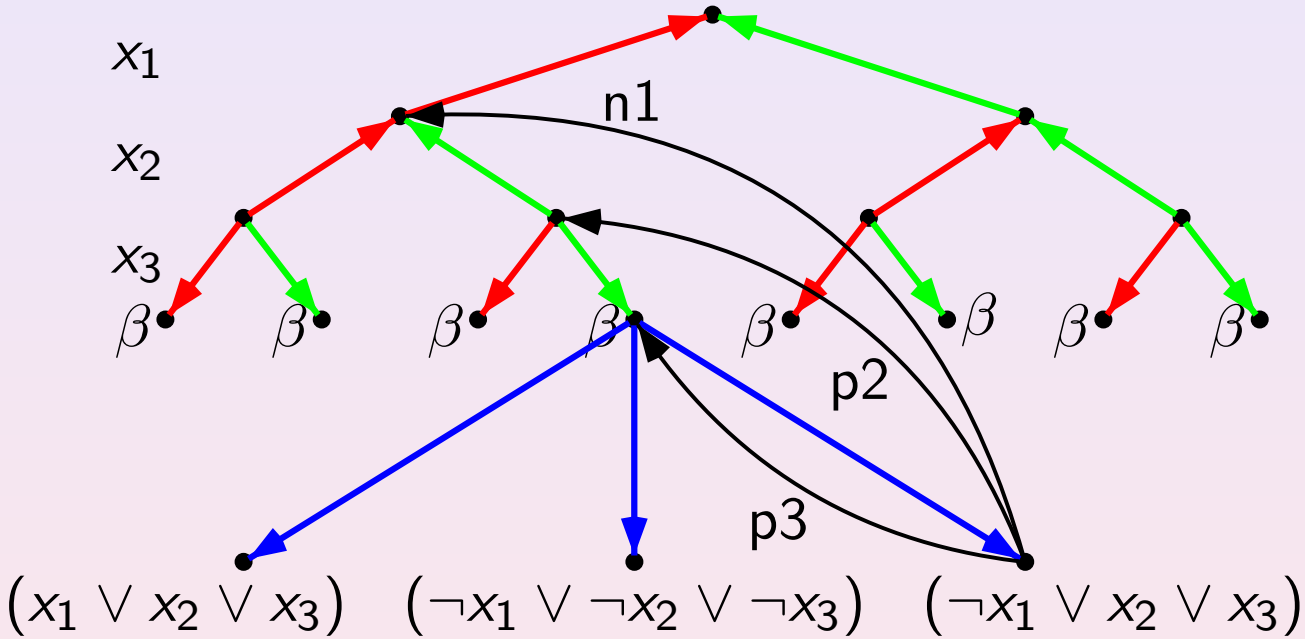


$$\exists x : \beta(x) \wedge \forall y : x \longrightarrow y \Rightarrow \bigvee_{i=1}^{3} \exists z, z' : y \xrightarrow{\text{pi}} z \longrightarrow z' \vee y \xrightarrow{\text{ni}} z \longrightarrow z'$$

Data complexity of Σ_2-FO is NP-hard

Reduction from the NP-hard problem 3-SAT

Example: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$



$$\exists x : \beta(x) \wedge \forall y : x \longrightarrow y \Rightarrow \bigvee_{i=1}^{3} \forall z, z' : \neg(y \xrightarrow{\text{pi}} z \longrightarrow z') \wedge \neg(y \xrightarrow{\text{ni}} z \longrightarrow z')$$

Complexity of MSO

data complexity	non-hierarchical	c-bounded HGD	general HGD
Σ_k-MSO	Σ_k^{poly}		Σ_k^{exp}

Complexity of MSO

data complexity	non-hierarchical	c-bounded HGD	general HGD
Σ_k-MSO	Σ_k^{poly}		Σ_k^{exp}

combined complexity	non-hierarchical	c-bounded HGD	general HGD
Σ_k-MSO	Σ_k^{poly}	Σ_k^{exp}	

Complexity of MSO

data complexity	non-hierarchical	c-bounded HGD	general HGD
Σ_k-MSO	Σ_k^{poly}		Σ_k^{exp}

combined complexity	non-hierarchical	c-bounded HGD	general HGD
Σ_k-MSO	Σ_k^{poly}	Σ_k^{exp}	

Remark: The apex restriction does not help in the context of MSO.

Future Work/Open Problems

- Close the gap between NL and P for the data complexity of FO for c-bounded HGDs.

Future Work/Open Problems

- Close the gap between NL and P for the data complexity of FO for c-bounded HGDs.
- Under which conditions is hierarchical model-checking for a logic harder than "flat" model-checking?
- When can hierarchy be exploited?