Synthesising Recursive Functions for First-Order Model Counting

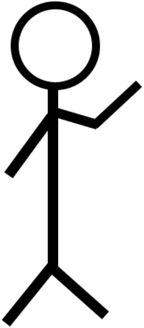
Paulius Dilkas

Joint work with Vaishak Belle (Univeristy of Edinburgh, UK)

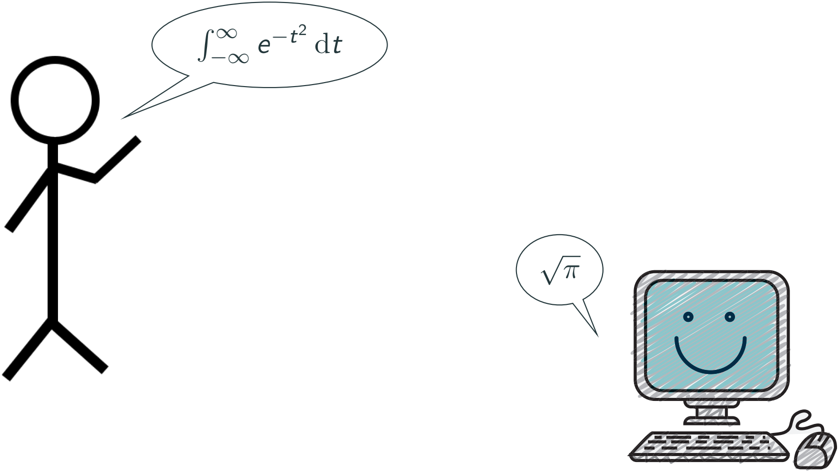
17th March 2023

National University of Singapore, Singapore

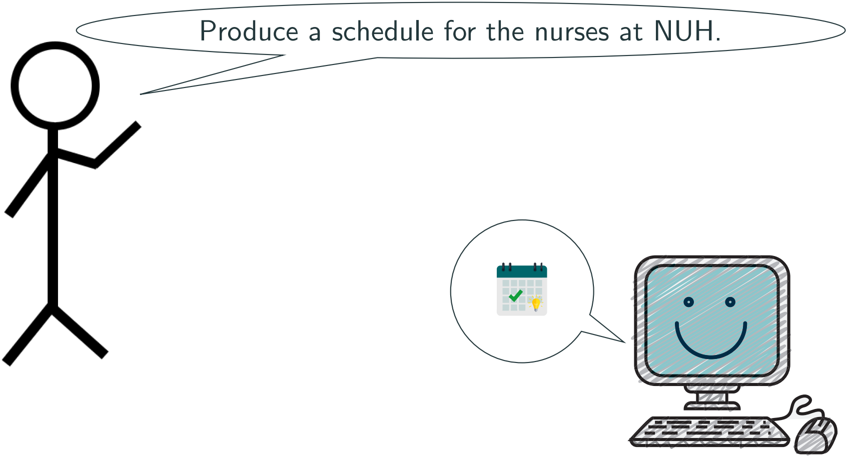
What Computers Can and Cannot Do



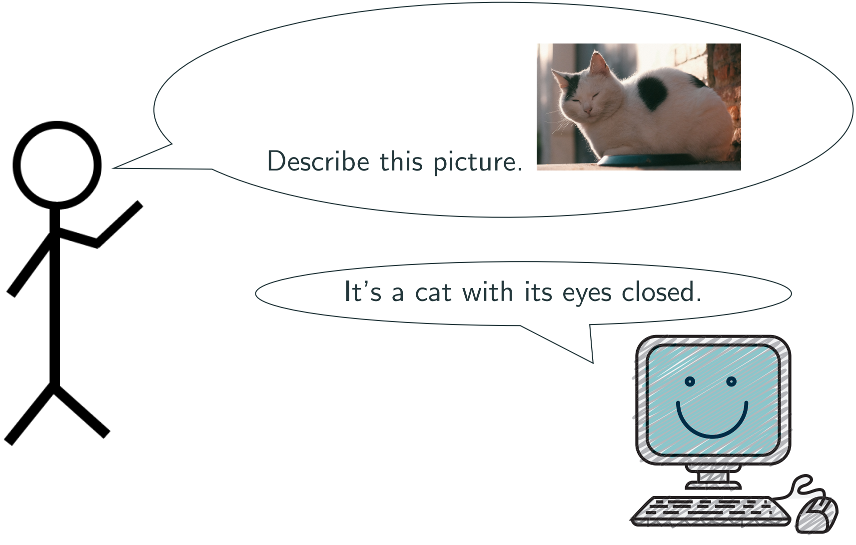
What Computers Can and Cannot Do



What Computers Can and Cannot Do

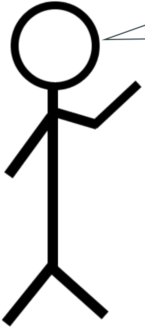


What Computers Can and Cannot Do

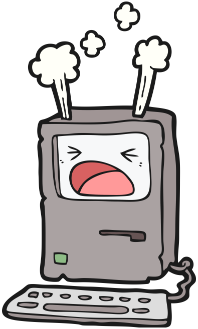


Vector art by Vecteezy.com

What Computers Can and Cannot Do



If I shuffle a deck of n cards,
how many possible outcomes
are there?



Terms and conditions apply.

Who Cares About Counting?

Who Cares About Counting?

Probabilistic Programming

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS,
DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON,
GERDA JANSSENS and LUC DE RAEDT

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(*e-mail:* `FirstName.LastName@cs.kuleuven.be`)

Who Cares About Counting?

Probabilistic Programming

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS, DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON, GERDA JANSSENS and LUC DE RAEDT

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(*e-mail:* `FirstName.LastName@cs.kuleuven.be`)

Neuro-symbolic AI

A Semantic Loss Function for Deep Learning with Symbolic Knowledge

Jingyi Xu [1] **Zilu Zhang** [2] **Tal Friedman** [1] **Yitao Liang** [1] **Guy Van den Broeck** [1]

Who Cares About Counting?

Probabilistic Programming

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS, DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON, GERDA JANSSENS and LUC DE RAEDT

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(*e-mail:* `FirstName.LastName@cs.kuleuven.be`)

Neuro-symbolic AI

A Semantic Loss Function for Deep Learning with Symbolic Knowledge

Jingyi Xu [1] **Zilu Zhang** [2] **Tal Friedman** [1] **Yitao Liang** [1] **Guy Van den Broeck** [1]

Natural Language Processing

Joint Inference for Knowledge Extraction from Biomedical Literature

Hoifung Poon[*]
Dept. of Computer Sci. & Eng.
University of Washington
Seattle, WA 98195
`hoifung@cs.washington.edu`

Lucy Vanderwende
Microsoft Research
Redmond, WA 98052
`Lucy.Vanderwende@microsoft.com`

Who Cares About Counting?

Probabilistic Programming

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS, DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON, GERDA JANSSENS and LUC DE RAEDT

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(*e-mail:* FirstName.LastName@cs.kuleuven.be)

Neuro-symbolic AI

A Semantic Loss Function for Deep Learning with Symbolic Knowledge

Jingyi Xu [1] **Zilu Zhang** [2] **Tal Friedman** [1] **Yitao Liang** [1] **Guy Van den Broeck** [1]

Natural Language Processing

Joint Inference for Knowledge Extraction from Biomedical Literature

Hoifung Poon*
Dept. of Computer Sci. & Eng.
University of Washington
Seattle, WA 98195
hoifung@cs.washington.edu

Lucy Vanderwende
Microsoft Research
Redmond, WA 98052
Lucy.Vanderwende@microsoft.com

Robotics

Learning Relational Affordance Models for Robots in Multi-Object Manipulation Tasks

Bogdan Moldovan Plinio Moreno Martijn van Otterlo José Santos-Victor Luc De Raedt

Who Cares About Counting?

Probabilistic Programming

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS, DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON, GERDA JANSSENS and LUC DE RAEDT

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(*e-mail:* FirstName.LastName@cs.kuleuven.be)

Neuro-symbolic AI

A Semantic Loss Function for Deep Learning with Symbolic Knowledge

Jingyi Xu [1] **Zilu Zhang** [2] **Tal Friedman** [1] **Yitao Liang** [1] **Guy Van den Broeck** [1]

Natural Language Processing

Joint Inference for Knowledge Extraction from Biomedical Literature

Hoifung Poon*
Dept. of Computer Sci. & Eng.
University of Washington
Seattle, WA 98195
hoifung@cs.washington.edu

Lucy Vanderwende
Microsoft Research
Redmond, WA 98052
Lucy.Vanderwende@microsoft.com

Robotics

Learning Relational Affordance Models for Robots in Multi-Object Manipulation Tasks

Bogdan Moldovan Plinio Moreno Martijn van Otterlo José Santos-Victor Luc De Raedt

Bioinformatics

PheNetic: Network-based interpretation of unstructured gene lists in E. coli

Dries De Maeyer[1], Joris Renkens[2], Lore Cloots[1], Luc De Raedt[2], Kathleen Marchal[1,3]

[1]Center of Microbial and Plant Genetics, Kasteelpark Arenberg 20, B-3001, Leuven, Belgium
[2]Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium
[3]Department of Plant Biotechnology and Bioinformatics, Ghent University, Technologiepark 927, 9052 Gent, Belgium

Who Cares About Counting?

Probabilistic Programming

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS, DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON, GERDA JANSSENS and LUC DE RAEDT

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(*e-mail:* FirstName.LastName@cs.kuleuven.be)

Neuro-symbolic AI

A Semantic Loss Function for Deep Learning with Symbolic Knowledge

Jingyi Xu [1] **Zilu Zhang** [2] **Tal Friedman** [1] **Yitao Liang** [1] **Guy Van den Broeck** [1]

Natural Language Processing

Joint Inference for Knowledge Extraction from Biomedical Literature

Hoifung Poon*
Dept. of Computer Sci. & Eng.
University of Washington
Seattle, WA 98195
hoifung@cs.washington.edu

Lucy Vanderwende
Microsoft Research
Redmond, WA 98052
Lucy.Vanderwende@microsoft.com

Robotics

Learning Relational Affordance Models for Robots in Multi-Object Manipulation Tasks

Bogdan Moldovan Plinio Moreno Martijn van Otterlo José Santos-Victor Luc De Raedt

Bioinformatics

PheNetic: Network-based interpretation of unstructured gene lists in E. coli

Dries De Maeyer[1], Joris Renkens[2], Lore Cloots[1], Luc De Raedt[2], Kathleen Marchal[1,3]

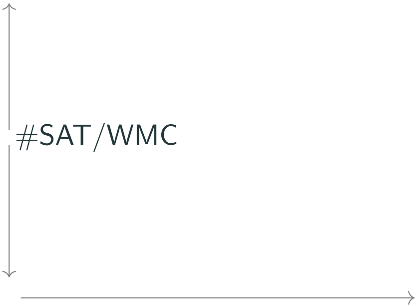
[1]Center of Microbial and Plant Genetics, Kasteelpark Arenberg 20, B-3001, Leuven, Belgium
[2]Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium
[3]Department of Plant Biotechnology and Bioinformatics, Ghent University, Technologiepark 927, 9052 Gent, Belgium

Combinatorics

Automatic Conjecturing of P-Recursions Using Lifted Inference

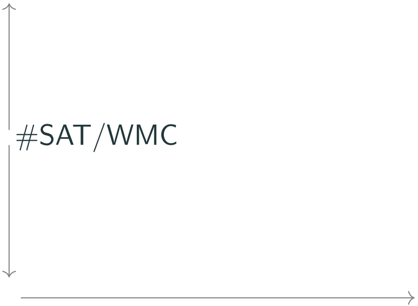
Jáchym Barvínek[1(✉)], Timothy van Bremen[2], Yuyi Wang[3], Filip Železný[1], and Ondřej Kuželka[1]

[1] Czech Technical University in Prague, Prague, Czech Republic
barvijac@fel.cvut.cz
[2] KU Leuven, Leuven, Belgium
[3] ETH Zurich, Zurich, Switzerland



#SAT (Valiant 1979)

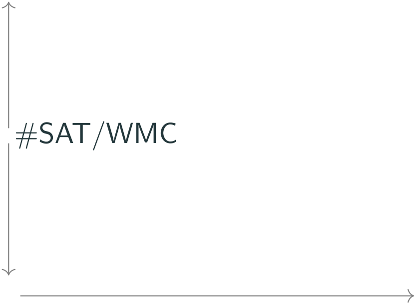
- Input formula: $x \vee y$
- Interpretations: $\emptyset$, $\{x\}$, $\{y\}$, $\{x, y\}$
- Models: $\{x\}$, $\{y\}$, $\{x, y\}$



#SAT **(Valiant 1979)**

- Input formula: $x \lor y$
- Interpretations: $\emptyset$, $\{x\}$, $\{y\}$, $\{x, y\}$
- Models: $\{x\}$, $\{y\}$, $\{x, y\}$
- Answer (model count): 3

(Some of the) Many Ways to Count

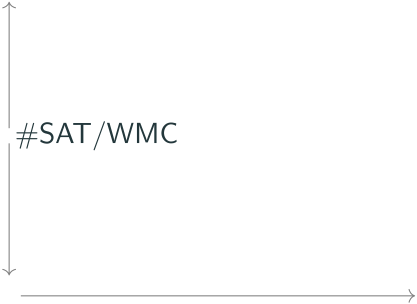


Weighted Model Counting **(Chavira and Darwiche 2008)**

- Input formula: $x \vee y$
- Input weights: $w(x) = 0.3$, $w(\neg x) = 0.7$,

  $w(y) = 0.2$, $w(\neg y) = 0.8$

(Some of the) Many Ways to Count



Weighted Model Counting **(Chavira and Darwiche 2008)**

- Input formula: $x \vee y$
- Input weights: $w(x) = 0.3$, $w(\neg x) = 0.7$,

  $w(y) = 0.2$, $w(\neg y) = 0.8$
- Answer (weighted model count):

  $w(x)w(y) + w(x)w(\neg y) + w(\neg x)w(y) = 0.44$

(Some of the) Many Ways to Count

#SAT/WMC $\longrightarrow$ (W)FOMC

From propositional to first-order logic

(Weighted) (Symmetric) First-Order Model Counting (Van den Broeck et al. 2011)

- Input formula: $\forall x \in \Delta.\ \mathtt{P}(x)$
- Input weights: $w^{+}(\mathtt{P}) = 0.3$, $w^{-}(\mathtt{P}) = 0.7$
- Input domain size(s): $|\Delta| = 2$

(Some of the) Many Ways to Count

#SAT/WMC $\longrightarrow$ (W)FOMC

From propositional to first-order logic

(Weighted) (Symmetric) First-Order Model Counting

(Van den Broeck et al. 2011)

- Input formula: $\forall x \in \Delta.\ \mathtt{P}(x)$
- Input weights: $w^{+}(\mathtt{P}) = 0.3$, $w^{-}(\mathtt{P}) = 0.7$
- Input domain size(s): $|\Delta| = 2$
- Answer: $(w^{+}(\mathtt{P}))^{|\Delta|} = 0.09$

(Some of the) Many Ways to Count

$$\begin{tikzcd} \text{WMI} & \text{WFOMI} \\ \#\text{SAT/WMC} \arrow[u] \arrow[r] & \text{(W)FOMC} \arrow[u] \end{tikzcd}$$

$\sum \rightarrow \int$

From propositional to first-order logic

Extensions to Continuous Domains

- Weighted model integration
  - (Belle, Passerini and Van den Broeck 2015)
- Weighted first-order model integration
  - (Feldstein and Belle 2021)

(Some of the) Many Ways to Count

$\sum \to \int$

$$\begin{tikzcd}
\text{WMI} & \text{WFOMI} \\
\#\text{SAT}/\text{WMC} \arrow[u] \arrow[r] \arrow[d] & \text{(W)FOMC} \arrow[u] \arrow[d] \\
\text{AMC}, \text{PBP} & \text{SP}
\end{tikzcd}$$

Generalising weights

From propositional to first-order logic

Generalisations of the weight function

- Algebraic model counting
  - (Kimmig, Van den Broeck and De Raedt 2017)
  - From $\mathbb{R}_{\geq 0}$ to commutative semirings
- Pseudo-Boolean projection (D. and Belle 2021)
  - Weights not necessarily on literals
- Semiring programming (Belle and De Raedt 2020)

(Some of the) Many Ways to Count

$\sum \to \int$

$$\begin{tikzcd}
\text{WMI} & \text{WFOMI} \\
\#\text{SAT}/\text{WMC} \arrow[u] \arrow[d] \arrow[r] & \text{(W)FOMC} \arrow[u] \arrow[d] \\
\text{AMC}, \text{PBP} & \text{SP}
\end{tikzcd}$$

Generalising weights

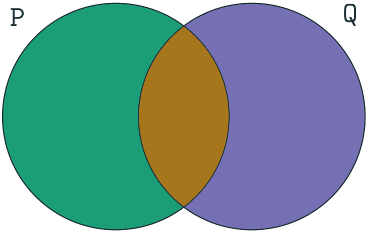
From propositional to first-order logic

(Unweighted) First-Order Model Counting

- Example formula:
  $\forall x \in \Delta.\ \mathtt{P}(x) \lor \mathtt{Q}(x)$
- Let $\Delta := \{ 1, 2 \}$
- Interpretations: all subsets of
  $\{ \mathtt{P}(1), \mathtt{Q}(1), \mathtt{P}(2), \mathtt{Q}(2) \}$

(Unweighted) First-Order Model Counting

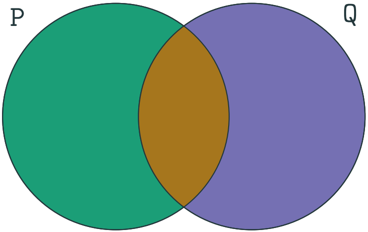
- Example formula:
  $\forall x \in \Delta.\ \mathtt{P}(x) \vee \mathtt{Q}(x)$
- Let $\Delta \coloneqq \{ 1, 2 \}$
- Interpretations: all subsets of
  $\{ \mathtt{P}(1), \mathtt{Q}(1), \mathtt{P}(2), \mathtt{Q}(2) \}$
- Models:



$$\{ \mathtt{P}(1), \mathtt{P}(2) \}, \quad \{ \mathtt{P}(1), \mathtt{Q}(2) \}, \quad \{ \mathtt{P}(1), \mathtt{P}(2), \mathtt{Q}(2) \},$$
$$\{ \mathtt{Q}(1), \mathtt{P}(2) \}, \quad \{ \mathtt{Q}(1), \mathtt{Q}(2) \}, \quad \{ \mathtt{Q}(1), \mathtt{P}(2), \mathtt{Q}(2) \},$$
$$\{ \mathtt{P}(1), \mathtt{Q}(1), \mathtt{P}(2) \}, \quad \{ \mathtt{P}(1), \mathtt{Q}(1), \mathtt{Q}(2) \}, \quad \{ \mathtt{P}(1), \mathtt{Q}(1), \mathtt{P}(2), \mathtt{Q}(2) \}$$

(Unweighted) First-Order Model Counting

- Example formula:
  $\forall x \in \Delta.\ \mathtt{P}(x) \lor \mathtt{Q}(x)$
- Let $\Delta \coloneqq \{ 1, 2 \}$
- Interpretations: all subsets of
  $\{ \mathtt{P}(1), \mathtt{Q}(1), \mathtt{P}(2), \mathtt{Q}(2) \}$
- Models:



$$\begin{array}{ccc}
\{ \mathtt{P}(1), \mathtt{P}(2) \}, & \{ \mathtt{P}(1), \mathtt{Q}(2) \}, & \{ \mathtt{P}(1), \mathtt{P}(2), \mathtt{Q}(2) \}, \\
\{ \mathtt{Q}(1), \mathtt{P}(2) \}, & \{ \mathtt{Q}(1), \mathtt{Q}(2) \}, & \{ \mathtt{Q}(1), \mathtt{P}(2), \mathtt{Q}(2) \}, \\
\{ \mathtt{P}(1), \mathtt{Q}(1), \mathtt{P}(2) \}, & \{ \mathtt{P}(1), \mathtt{Q}(1), \mathtt{Q}(2) \}, & \{ \mathtt{P}(1), \mathtt{Q}(1), \mathtt{P}(2), \mathtt{Q}(2) \}
\end{array}$$

Intuition

- Each 1-ary predicate is like a subset
- For $n > 1$, each n-ary predicate is like a relation
- FOMC counts combinations of relations

First-Order Knowledge Compilation: An Approach to FOMC

$$\forall x \in \Delta.\ \mathtt{P}(x) \vee \mathtt{Q}(x)$$

First-Order Knowledge Compilation: An Approach to FOMC

$$\forall x \in \Delta.\ \mathtt{P}(x) \lor \mathtt{Q}(x)$$

Independent partial grounding (introduces a constant $c \in \Delta$)

First-Order Knowledge Compilation: An Approach to FOMC

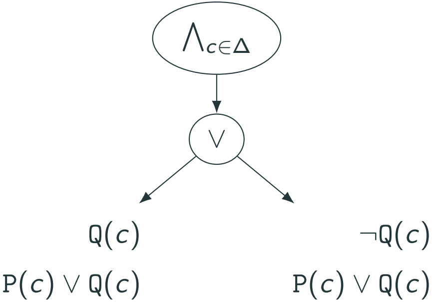
$$\begin{tikzcd} \bigwedge_{c \in \Delta} \arrow[d] \\ \mathtt{P}(c) \lor \mathtt{Q}(c) \end{tikzcd}$$

Independent partial grounding (introduces a constant $c \in \Delta$)

$$\begin{array}{c}\bigwedge_{c \in \Delta} \\ \downarrow \\ \mathtt{P}(c) \vee \mathtt{Q}(c)\end{array}$$

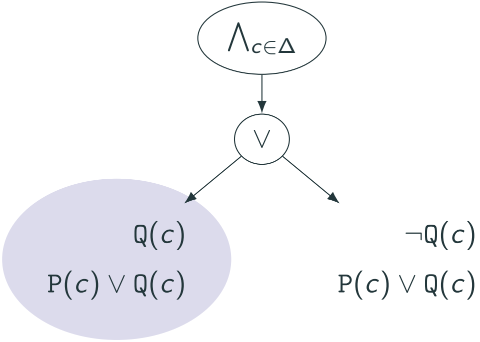
Shannon decomposition (a.k.a. Boole's expansion theorem) on $\mathtt{Q}(c)$

First-Order Knowledge Compilation: An Approach to FOMC

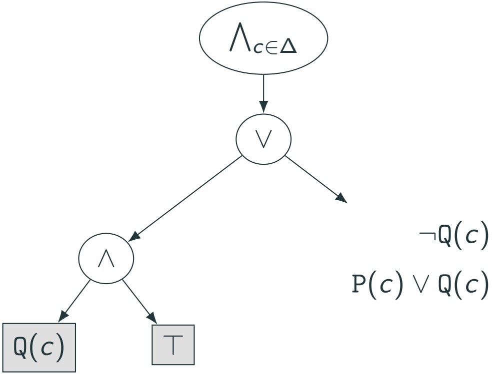


Shannon decomposition (a.k.a. Boole's expansion theorem) on $\mathtt{Q}(c)$

First-Order Knowledge Compilation: An Approach to FOMC

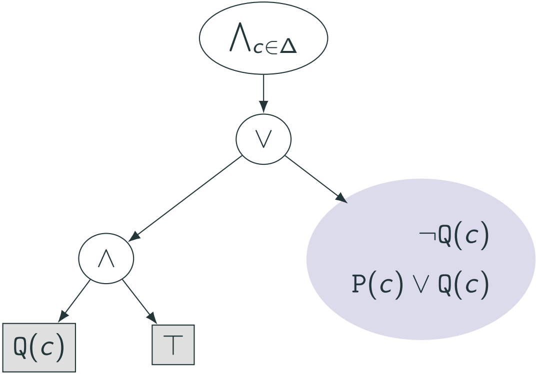


Positive unit propagation of $\mathtt{Q}(c)$



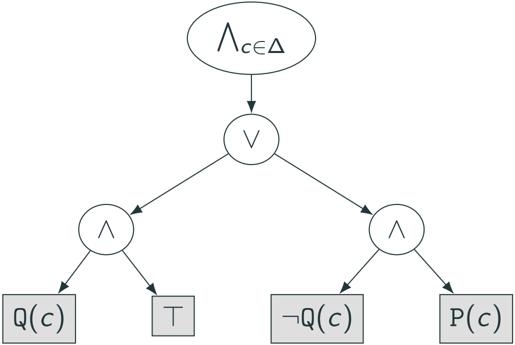
Positive unit propagation of $\mathtt{Q}(c)$

First-Order Knowledge Compilation: An Approach to FOMC



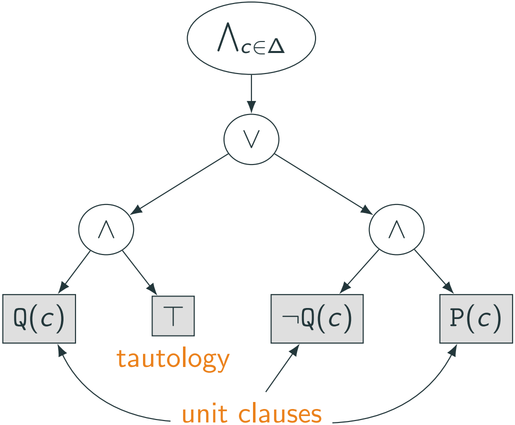
Negative unit propagation of $\neg\mathtt{Q}(c)$

First-Order Knowledge Compilation: An Approach to FOMC



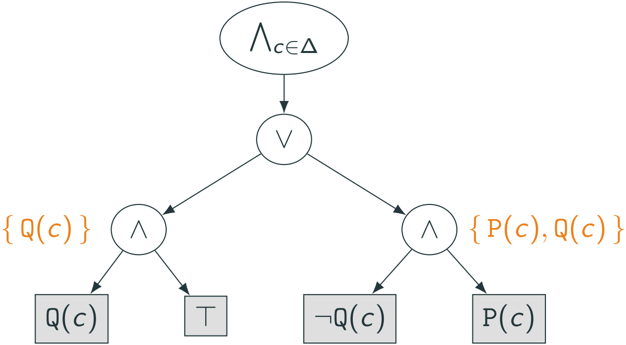
Negative unit propagation of $\neg\mathtt{Q}(c)$

First-Order Knowledge Compilation: An Approach to FOMC



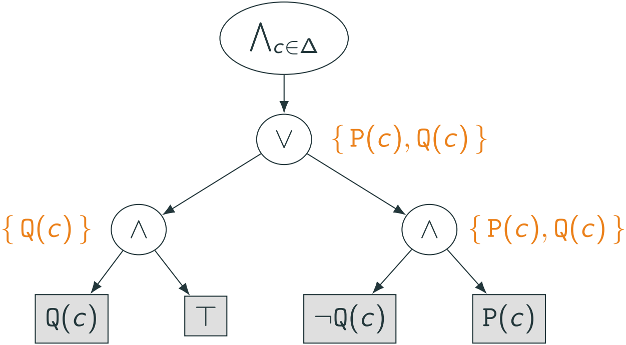
Compilation is complete ✓

First-Order Knowledge Compilation: An Approach to FOMC



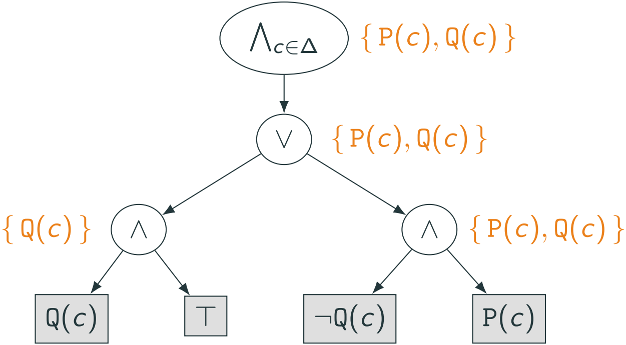
Smoothing: propagating atoms upwards

First-Order Knowledge Compilation: An Approach to FOMC



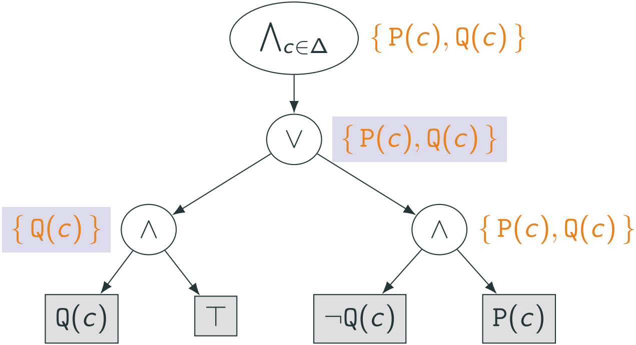
Smoothing: propagating atoms upwards

First-Order Knowledge Compilation: An Approach to FOMC



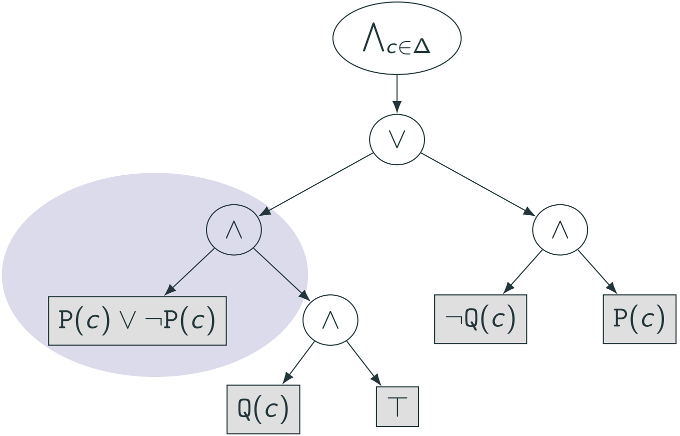
Smoothing: propagating atoms upwards

First-Order Knowledge Compilation: An Approach to FOMC



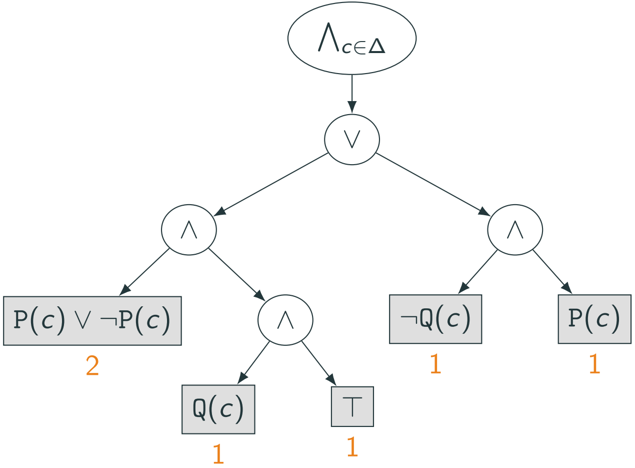
Smoothing: adding new atoms

First-Order Knowledge Compilation: An Approach to FOMC



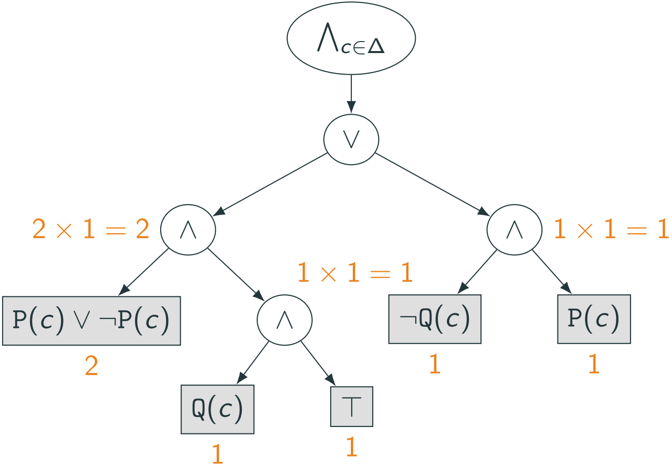
Smoothing: adding new atoms

First-Order Knowledge Compilation: An Approach to FOMC



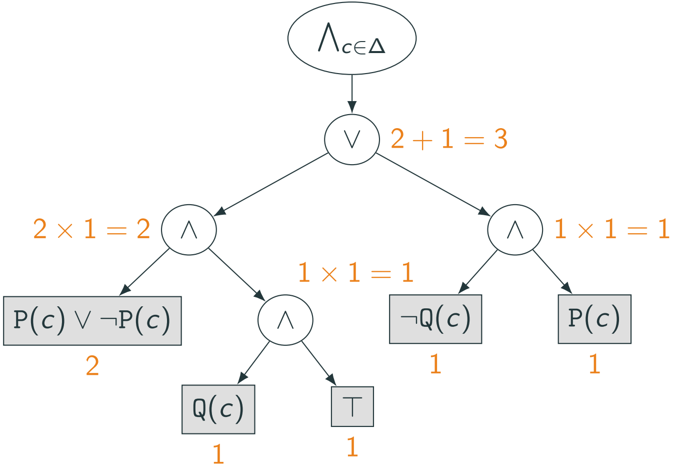
Propagating the model count

First-Order Knowledge Compilation: An Approach to FOMC

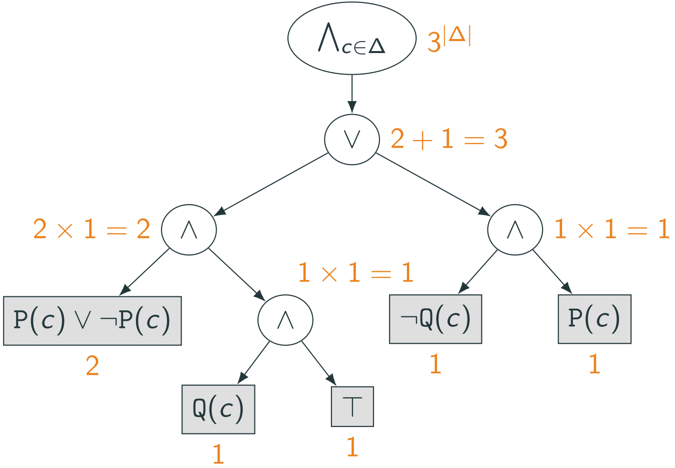


Propagating the model count

First-Order Knowledge Compilation: An Approach to FOMC



Propagating the model count



Propagating the model count

A (Slightly) More Complicated Example

Suppose this room has n seats, and there are $m \leq n$ people in the audience. How many ways are there to seat everyone?

A (Slightly) More Complicated Example

Suppose this room has n seats, and there are $m \leq n$ people in the audience. How many ways are there to seat everyone?

More explicitly, we assume that:

- each attendee gets exactly one seat,
- and a seat can accommodate at most one person.

A (Slightly) More Complicated Example

Suppose this room has n seats, and there are $m \leq n$ people in the audience. How many ways are there to seat everyone?

More explicitly, we assume that:

- each attendee gets exactly one seat,
- and a seat can accommodate at most one person.

Answer: $n^{\underline{m}} = n \cdot (n-1) \cdots (n-m+1)$.

Note: this problem is equivalent to counting $[m] \to [n]$ injections.

Let's Express This Problem in Logic!

- Let Γ and Δ be sets (i.e., domains)
  - such that $|\Gamma| = m$, and $|\Delta| = n$.
- Let $\mathtt{P} \subseteq \Gamma \times \Delta$ be a relation (i.e., predicate) over Γ and Δ.
- We can describe all of the constraints in first-order logic:

Let's Express This Problem in Logic!

- Let Γ and Δ be sets (i.e., domains)
  - such that $|\Gamma| = m$, and $|\Delta| = n$.
- Let $\mathtt{P} \subseteq \Gamma \times \Delta$ be a relation (i.e., predicate) over Γ and Δ.
- We can describe all of the constraints in first-order logic:
  - each attendee gets a seat (i.e., at least one seat)

$$\forall x \in \Gamma.\ \exists y \in \Delta.\ \mathtt{P}(x, y) \tag{1}$$

Let's Express This Problem in Logic!

- Let Γ and Δ be sets (i.e., domains)
    - such that $|\Gamma| = m$, and $|\Delta| = n$.
- Let $\mathtt{P} \subseteq \Gamma \times \Delta$ be a relation (i.e., predicate) over Γ and Δ.
- We can describe all of the constraints in first-order logic:
    - each attendee gets a seat (i.e., at least one seat)

$$\forall x \in \Gamma.\ \exists y \in \Delta.\ \mathtt{P}(x, y) \tag{1}$$

    - one person cannot occupy multiple seats

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ \mathtt{P}(x, y) \land \mathtt{P}(x, z) \Rightarrow y = z \tag{2}$$

Let's Express This Problem in Logic!

- Let Γ and Δ be sets (i.e., domains)
  - such that $|\Gamma| = m$, and $|\Delta| = n$.
- Let $\mathtt{P} \subseteq \Gamma \times \Delta$ be a relation (i.e., predicate) over Γ and Δ.
- We can describe all of the constraints in first-order logic:
  - each attendee gets a seat (i.e., at least one seat)

$$\forall x \in \Gamma.\ \exists y \in \Delta.\ \mathtt{P}(x, y) \tag{1}$$

  - one person cannot occupy multiple seats

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ \mathtt{P}(x, y) \land \mathtt{P}(x, z) \Rightarrow y = z \tag{2}$$

  - one seat cannot accommodate multiple attendees

$$\forall w, x \in \Gamma.\ \forall y \in \Delta.\ \mathtt{P}(w, y) \land \mathtt{P}(x, y) \Rightarrow w = x \tag{3}$$

Let's Express This Problem in Logic!

- Let Γ and Δ be sets (i.e., domains)
  - such that $|\Gamma| = m$, and $|\Delta| = n$.
- Let $\mathtt{P} \subseteq \Gamma \times \Delta$ be a relation (i.e., predicate) over Γ and Δ.
- We can describe all of the constraints in first-order logic:
  - each attendee gets a seat (i.e., at least one seat)

$$\forall x \in \Gamma.\ \exists y \in \Delta.\ \mathtt{P}(x, y) \tag{1}$$

  - one person cannot occupy multiple seats

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ \mathtt{P}(x, y) \wedge \mathtt{P}(x, z) \Rightarrow y = z \tag{2}$$

  - one seat cannot accommodate multiple attendees

$$\forall w, x \in \Gamma.\ \forall y \in \Delta.\ \mathtt{P}(w, y) \wedge \mathtt{P}(x, y) \Rightarrow w = x \tag{3}$$

(1) and (2) constrain $\mathtt{P}$ to be a function, and (3) makes it injective.

Recursion

Back to Our Example

The following function counts injections:

$$f(m, n) = \begin{cases} 1 & \text{if } m = 0 \text{ and } n = 0 \\ 0 & \text{if } m > 0 \text{ and } n = 0 \\ f(m, n-1) + mf(m-1, n-1) & \text{otherwise.} \end{cases}$$

Back to Our Example

The following function counts injections:

$$f(m,n) = \begin{cases} 1 & \text{if } m = 0 \text{ and } n = 0 \\ 0 & \text{if } m > 0 \text{ and } n = 0 \\ f(m, n-1) + mf(m-1, n-1) & \text{otherwise.} \end{cases}$$

- $f(m,n)$ can be computed in $\Theta(mn)$ time
  - using dynamic programming

Back to Our Example

The following function counts injections:

$$f(m,n) = \begin{cases} 1 & \text{if } m = 0 \text{ and } n = 0 \\ 0 & \text{if } m > 0 \text{ and } n = 0 \\ f(m, n-1) + mf(m-1, n-1) & \text{otherwise.} \end{cases}$$

- $f(m,n)$ can be computed in $\Theta(mn)$ time
  - using dynamic programming
- Optimal time complexity to compute $n^{\underline{m}}$ is $\Theta(m)$
- But $\Theta(mn)$ is still much better than translating to propositional logic and running a WMC algorithm

Back to Our Example

The following function counts injections:

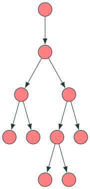
$$f(m,n) = \begin{cases} 1 & \text{if } m = 0 \text{ and } n = 0 \\ 0 & \text{if } m > 0 \text{ and } n = 0 \\ f(m,n-1) + mf(m-1,n-1) & \text{otherwise.} \end{cases}$$

- $f(m,n)$ can be computed in $\Theta(mn)$ time
  - using dynamic programming
- Optimal time complexity to compute $n^{\underline{m}}$ is $\Theta(m)$
- But $\Theta(mn)$ is still much better than translating to propositional logic and running a WMC algorithm
- The rest of this talk is about how to find such functions automatically

First-Order Knowledge Compilation: Before and After

$\forall x \in \Delta.\ \mathtt{P}(x) \vee \mathtt{Q}(x)$

Compilation



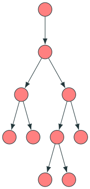
Propagation ← $|\Delta| = 4$

81

First-Order Knowledge Compilation: Before and After

$\forall x \in \Delta.\ \mathtt{P}(x) \lor \mathtt{Q}(x)$

Compilation



Propagation ← $|\Delta| = 4$

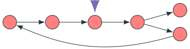
81

$\forall x \in \Gamma.\ \exists y \in \Delta.\ \mathtt{P}(x, y)$

$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ \mathtt{P}(x, y) \land \mathtt{P}(x, z) \Rightarrow y = z$

$\forall w, x \in \Gamma.\ \forall y \in \Delta.\ \mathtt{P}(w, y) \land \mathtt{P}(x, y) \Rightarrow w = x$

Compilation



Conversion

$$f(m, n) = \sum_{l=0}^{m} \binom{m}{l} [l < 2] \times f(m - l, n - 1)$$

Simplification

$$f(m, n) = f(m, n - 1) + mf(m - 1, n - 1)$$

Evaluation ← $f(0, 0) = 1,\ f(m, 0) = 0$; $m = 2,\ n = 7$

42

Circuits vs Graphs

Circuits (Van den Broeck et al. 2011)...

- ...extend d-DNNF circuits (Darwiche 2001) for propositional knowledge compilation with more node types
- ...are acyclic.

Circuits vs Graphs

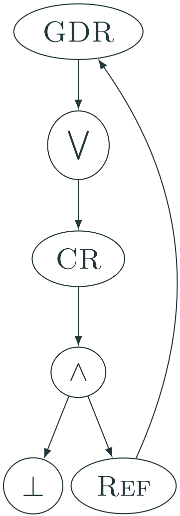
Circuits (Van den Broeck et al. 2011). . .

- . . . extend d-DNNF circuits (Darwiche 2001) for propositional knowledge compilation with more node types
- . . . are acyclic.

First-Order Computational Graphs (FCGs) are. . .
directed ~~acyclic~~ (weakly connected) graphs with:

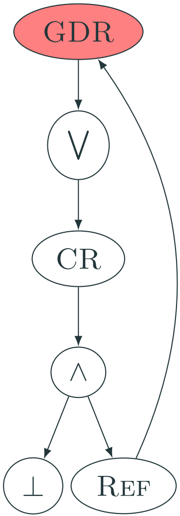
- a single source,
- labelled nodes,
- and ordered outgoing edges.

How to Interpret an FCG



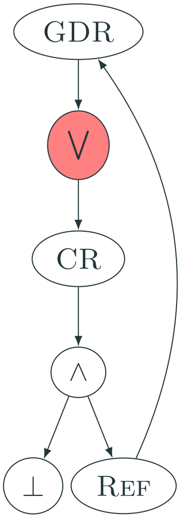
$f(m, n) =$

How to Interpret an FCG



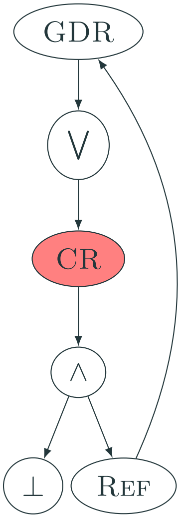
$$f(m, n) =$$

How to Interpret an FCG



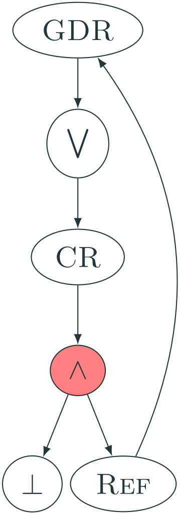
$$f(m, n) = \sum_{l=0}^{m} \binom{m}{l}$$

How to Interpret an FCG



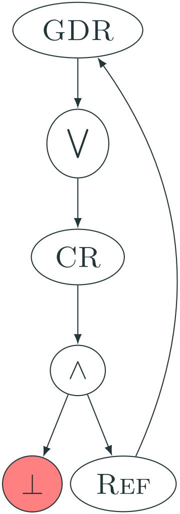
$$f(m,n) = \sum_{l=0}^{m} \binom{m}{l}$$

How to Interpret an FCG



$$f(m,n) = \sum_{l=0}^{m} \binom{m}{l} \qquad \times$$

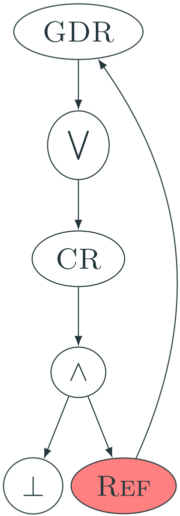
How to Interpret an FCG



$$f(m, n) = \sum_{l=0}^{m} \binom{m}{l} [l < 2] \times$$

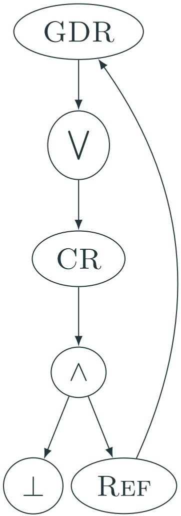
$$[\phi] = \begin{cases} 1 & \text{if } \phi \\ 0 & \text{if } \neg\phi \end{cases}$$

How to Interpret an FCG



$$f(m,n) = \sum_{l=0}^{m} \binom{m}{l} [l < 2] \times f(m-l, n-1)$$

How to Interpret an FCG



$$
\begin{aligned}
f(m,n) &= \sum_{l=0}^{m} \binom{m}{l} [l < 2] \times f(m-l, n-1) \\
&= f(m, n-1) + mf(m-1, n-1)
\end{aligned}
$$

Definition
A (compilation) rule is a function that takes a formula and returns a set of (G, L) pairs, where

- G is an FCG,
- and L is a list of formulas.

Example Rule: Independence

Input formula:

$$(\forall x, y \in \Omega.\ x = y) \wedge \tag{1}$$

$$(\forall x \in \Gamma.\ \forall y, z \in \Delta.\ \mathtt{P}(x, y) \wedge \mathtt{P}(x, z) \Rightarrow y = z) \wedge \tag{2}$$

$$(\forall w, x \in \Gamma.\ \forall y \in \Delta.\ \mathtt{P}(w, y) \wedge \mathtt{P}(x, y) \Rightarrow w = x) \tag{3}$$

Example Rule: Independence

Input formula:

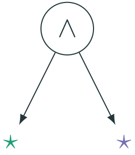
$$(\forall x, y \in \Omega.\ x = y) \wedge \tag{1}$$

$$(\forall x \in \Gamma.\ \forall y, z \in \Delta.\ \mathtt{P}(x, y) \wedge \mathtt{P}(x, z) \Rightarrow y = z) \wedge \tag{2}$$

$$(\forall w, x \in \Gamma.\ \forall y \in \Delta.\ \mathtt{P}(w, y) \wedge \mathtt{P}(x, y) \Rightarrow w = x) \tag{3}$$

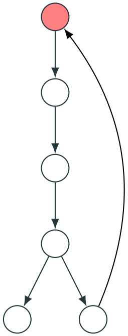
Only one (G, L) pair:

$G =$



$, \qquad L = \langle (1), (2) \wedge (3) \rangle$

New Rule 1: Generalised Domain Recursion



Example

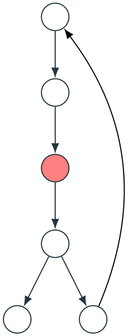
Input formula:

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ y \neq z \Rightarrow \neg \mathrm{P}(x, y) \vee \neg \mathrm{P}(x, z)$$

Output formula (with a new constant $c \in \Gamma$):

$$\forall y, z \in \Delta.\ y \neq z \Rightarrow \neg \mathrm{P}(c, y) \vee \neg \mathrm{P}(c, z)$$

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ x \neq c \wedge y \neq z \Rightarrow \neg \mathrm{P}(x, y) \vee \neg \mathrm{P}(x, z)$$



Example

Input formula (with a constant $c \in \Gamma$):

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ x \neq c \land y \neq z \Rightarrow \neg \mathrm{P}(x, y) \lor \neg \mathrm{P}(x, z)$$

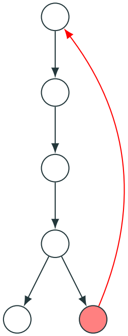
$$\forall w, x \in \Gamma.\ \forall y \in \Delta.\ w \neq c \land x \neq c \land w \neq x \Rightarrow \neg \mathrm{P}(w, y) \lor \neg \mathrm{P}(x, y)$$

Output formula (with a new domain $\Gamma' := \Gamma \setminus \{ c \}$):

$$\forall x \in \Gamma'.\ \forall y, z \in \Delta.\ y \neq z \Rightarrow \neg \mathrm{P}(x, y) \lor \neg \mathrm{P}(x, z)$$

$$\forall w, x \in \Gamma'.\ \forall y \in \Delta.\ w \neq x \Rightarrow \neg \mathrm{P}(w, y) \lor \neg \mathrm{P}(x, y)$$

New Rule 3: Identifying Possibilities for Recursion



Goal
Check if the input formula is isomorphic (up to domains) to a previously encountered formula.

Rough Outline

1. Consider pairs of 'similar' clauses.
2. Consider bijections between their sets of variables.
3. Extend each such bijection to a map between sets of domains.
4. If the bijection makes the clauses equal, and the domain map is compatible with previous domain maps, move on to another pair of clauses.

Resulting Improvements to Counting Functions

Let Γ and Δ be two sets with cardinalities $|\Gamma| = m$ and $|\Delta| = n$.

Our new rules enable $\textsc{ForcLift}$ to efficiently count $\Gamma \to \Delta$ functions such as:

- injections in $\Theta(mn)$ time
  - best: $\Theta(m)$
- partial injections in $\Theta(mn)$ time
  - best: $\Theta(\min\{\, m, n \,\}^2)$
- bijections in $\Theta(m)$ time
  - optimal!

Summary & Future Work

Summary

The circuits hitherto used for FOMC become more powerful with:

- cycles,
- generalised domain recursion,
- and some more new compilation rules that support domain recursion.

Future Work

- Automate:
  - extracting and simplifying the definitions of functions,
  - finding all base cases.
- Open questions:
  - What kind of sequences are computable in this way?
  - Would using a different logic extend the capabilities of FOMC further?