Towards Machine Learning for Quantification

Mikoláš Janota
AITP, 28 March 2018

IST/INESC-ID,
University of Lisbon, Portugal
Intro: QBF, Expansion, Games, Careful expansion

Solving QBF

Learning in QBF

Bernays–Schönfinkel (“Effectively Propositional Logic”) — Finite Models
Intro: QBF, Expansion, Games, Careful expansion
SAT and QBF

- **SAT** — for a Boolean formula, determine if it is **satisfiable**
• **SAT** — for a Boolean formula, determine if it is **satisfiable**

• Example: \( \{x = 1, y = 0\} \models (x \lor y) \land (x \lor \neg y) \)
SAT and QBF

• **SAT** — for a Boolean formula, determine if it is **satisfiable**
  
  Example: \( \{x = 1, y = 0\} \models (x \lor y) \land (x \lor \neg y) \)

• **QBF** — for a *Quantified* Boolean formula

  Quantifications as shorthands for connectives \( (\forall = \land, \exists = \lor) \)

  Example:
  
  1. \( (1) \forall x \exists y : (x \land y) \)
  2. \( (2) \forall x : (x \land 0) \lor (x \land 1) \)
  3. \( (3) ((0 \land 0) \lor (0 \land 1)) \land ((1 \land 0) \lor (1 \land 1)) \)
  4. \( (4) \top \)
SAT and QBF

- **SAT** — for a Boolean formula, determine if it is satisfiable

  Example: \( \{ x = 1, y = 0 \} \models (x \lor y) \land (x \lor \neg y) \)

- **QBF** — for a *Quantified* Boolean formula

  Example: \( \forall x \exists y. (x \leftrightarrow y) \)
SAT and QBF

• **SAT** — for a Boolean formula, determine if it is *satisfiable*

  Example: \( \{x = 1, y = 0\} \models (x \lor y) \land (x \lor \neg y) \)

• **QBF** — for a *Quantified* Boolean formula

  Example: \( \forall x \exists y. (x \leftrightarrow y) \)

• Quantifications as shorthands for connectives

  \( (\forall = \land, \exists = \lor) \)

  Example:
SAT and QBF

- **SAT** — for a Boolean formula, determine if it is **satisfiable**
- Example: \( \{x = 1, y = 0\} \models (x \lor y) \land (x \lor \lnot y) \)
- **QBF** — for a *Quantified* Boolean formula
- Example: \( \forall x \exists y. (x \leftrightarrow y) \)
- Quantifications as shorthands for connectives \( (\forall = \land, \exists = \lor) \)

**Example:**

1. \( \forall x \exists y. (x \leftrightarrow y) \)
• **SAT** — for a Boolean formula, determine if it is *satisfiable*  
  
  Example: \( \{x = 1, y = 0\} \models (x \lor y) \land (x \lor \neg y) \)

• **QBF** — for a *Quantified* Boolean formula  
  
  Example: \( \forall x \exists y. (x \leftrightarrow y) \)

• Quantifications as shorthands for connectives  
  
  \( \forall = \land, \exists = \lor \)  
  
  Example:  
  
  (1) \( \forall x \exists y. (x \leftrightarrow y) \)  
  
  (2) \( \forall x. (x \leftrightarrow 0) \lor (x \leftrightarrow 1) \)
SAT and QBF

- **SAT** — for a Boolean formula, determine if it is satisfiable
  - Example: \( \{x = 1, y = 0\} \models (x \lor y) \land (x \lor \neg y) \)

- **QBF** — for a *Quantified* Boolean formula
  - Example: \( \forall x \exists y. (x \leftrightarrow y) \)

- Quantifications as shorthands for connectives
  \( \forall = \land, \exists = \lor \)

Example:

1. \( \forall x \exists y. (x \leftrightarrow y) \)
2. \( \forall x. (x \leftrightarrow 0) \lor (x \leftrightarrow 1) \)
3. \( ((0 \leftrightarrow 0) \lor (0 \leftrightarrow 1)) \land ((1 \leftrightarrow 0) \lor (1 \leftrightarrow 1)) \)
SAT and QBF

- **SAT** — for a Boolean formula, determine if it is **satisfiable**
  - Example: \( \{x = 1, y = 0\} \models (x \lor y) \land (x \lor \neg y) \)
- **QBF** — for a *Quantified* Boolean formula
  - Example: \( \forall x \exists y. (x \leftrightarrow y) \)
- Quantifications as shorthands for connectives
  \( \forall = \land, \exists = \lor \)

Example:

1. \( \forall x \exists y. (x \leftrightarrow y) \)
2. \( \forall x. (x \leftrightarrow 0) \lor (x \leftrightarrow 1) \)
3. \( ((0 \leftrightarrow 0) \lor (0 \leftrightarrow 1)) \land ((1 \leftrightarrow 0) \lor (1 \leftrightarrow 1)) \)
4. \( 1 \) (True)
QBF is a strict subset of Bernays–Schönfinkel (EPR)

- Consider the QBF:
  \[ \forall u \exists e. u \leftrightarrow e \]
QBF is a strict subset of Bernays–Schönfinkel (EPR)

• Consider the QBF:

\[ \forall u \exists e. u \leftrightarrow e \]

1. Introduce a predicate for truth,
QBF is a strict subset of Bernays–Schönfinkel (EPR)

• Consider the QBF:

\[ \forall u \exists e. u \leftrightarrow e \]

1. Introduce a predicate for truth,
2. each existential variable replace by a predicate,
QBF is a strict subset of Bernays–Schönfinkel (EPR)

- Consider the QBF:

\[ \forall u \exists e. u \leftrightarrow e \]

1. Introduce a predicate for truth,
2. each existential variable replace by a predicate,
3. universal variables wrapped by the truth predicate:

\[
\text{is-true}(t) \land \neg \text{is-true}(f) \land \\
(\forall X_u. \text{is-true}(X_u) \leftrightarrow p_e(X_u))
\]
QBF is a strict subset of Bernays–Schönfinkel (EPR)

- Consider the QBF:
  \[ \forall u \exists e. \, u \leftrightarrow e \]

  1. Introduce a predicate for truth,
  2. each existential variable replace by a predicate,
  3. universal variables wrapped by the truth predicate:
     \[
     \text{is-true}(t) \land \neg \text{is-true}(f) \land \\
     (\forall X_u. \, \text{is-true}(X_u) \leftrightarrow p_e(X_u))
     \]

- Alternatively, use equality:
  \[
  t \neq f \land (\forall X_u. \, (X_u = t) \leftrightarrow p_e(X_u))
  \]
In this talk we consider prenex form:

Quantifier-prefix. Matrix

- A QBF represents a two-player game between 8 and 9.
- 8 wins a game if the matrix becomes false.
- 9 wins a game if the matrix becomes true.
- A QBF is false iff there exists a winning strategy for 8.
- A QBF is true iff there exists a winning strategy for 9.

Example 8 u 9 e:

9 -player wins by playing e, u.
• In this talk we consider prenex form:
  \[ \text{Quantifier-prefix. Matrix} \]

  Example \( \forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \lor e_1) \land (u_2 \lor \neg e_2) \)
In this talk we consider **prenex form**: Quantifier-prefix. Matrix

**Example** $\forall u_1u_2\exists e_1e_2. (\neg u_1 \lor e_1) \land (u_2 \lor \neg e_2)$

A QBF represents a two-player game between $\forall$ and $\exists$. 
Quantification and Two-player Games

- In this talk we consider **prenex form**: 
  Quantifier-prefix. Matrix

Example $\forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \lor e_1) \land (u_2 \lor \neg e_2)$

- A QBF represents a two-player game between $\forall$ and $\exists$.
- $\forall$ wins a game if the matrix becomes false.
Quantification and Two-player Games

- In this talk we consider **prenex form**: 
  \[ \text{Quantifier-prefix. Matrix} \]
  Example \( \forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \lor e_1) \land (u_2 \lor \neg e_2) \)
- A QBF represents a two-player game between \( \forall \) and \( \exists \).
- \( \forall \) wins a game if the matrix becomes false.
- \( \exists \) wins a game if the matrix becomes true.
Quantification and Two-player Games

• In this talk we consider prenex form: Quantifier-prefix. Matrix
  
  Example $\forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \lor e_1) \land (u_2 \lor \neg e_2)$

• A QBF represents a two-player game between $\forall$ and $\exists$.
  
  $\forall$ wins a game if the matrix becomes false.

  $\exists$ wins a game if the matrix becomes true.

• A QBF is false iff there exists a winning strategy for $\forall$. 
In this talk we consider prenex form: \( \text{Quantifier-prefix. Matrix} \)

Example: \( \forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \lor e_1) \land (u_2 \lor \neg e_2) \)

- A QBF represents a two-player game between \( \forall \) and \( \exists \).
- \( \forall \) wins a game if the matrix becomes false.
- \( \exists \) wins a game if the matrix becomes true.
- A QBF is false iff there exists a \textit{winning strategy} for \( \forall \).
- A QBF is true iff there exists a \textit{winning strategy} for \( \exists \).
• In this talk we consider **prenex form:**
  Quantifier-prefix. Matrix

Example $$\forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \lor e_1) \land (u_2 \lor \neg e_2)$$

• A QBF represents a two-player game between $$\forall$$ and $$\exists$$.
• $$\forall$$ wins a game if the matrix becomes false.
• $$\exists$$ wins a game if the matrix becomes true.
• A QBF is false iff there exists a **winning strategy** for $$\forall$$.
• A QBF is true iff there exists a **winning strategy** for $$\exists$$.

Example

$$\forall u \exists e. (u \leftrightarrow e)$$

$$\exists$$-player wins by playing $$e \Leftrightarrow u.$$
Solving QBF
Solving by CEGAR Expansion

\[ \exists \mathcal{E} \forall \mathcal{U}. \phi \equiv \exists \mathcal{E}. \bigwedge_{\mu \in 2^\mathcal{U}} \phi[\mu] \]
\[ \exists \mathcal{E} \forall \mathcal{U}. \phi \equiv \exists \mathcal{E}. \bigwedge_{\mu \in 2^\mathcal{U}} \phi[\mu] \]

Can be solved by SAT \( \bigwedge_{\mu \in 2^\mathcal{U}} \phi[\mu] \). Impractical!
\[ \exists \mathcal{E} \forall U. \phi \equiv \exists \mathcal{E}. \bigwedge_{\mu \in 2^U} \phi[\mu] \]

Can be solved by SAT \( \bigwedge_{\mu \in 2^U} \phi[\mu] \). Impractical!

Observe:

\[ \exists \mathcal{E}. \bigwedge_{\mu \in 2^U} \phi[\mu] \Rightarrow \exists \mathcal{E}. \bigwedge_{\mu \in \omega} \phi[\mu] \]

for some \( \omega \subseteq 2^U \)

What is a good \( \omega \)?
Solving by CEGAR Expansion Contd.

\[ \exists \mathcal{E} \forall U. \phi \equiv \exists \mathcal{E}. \bigwedge_{\mu \in 2^U} \phi[\mu] \]

Expand gradually instead: [Janota and Marques-Silva, 2011]

- Pick \( \tau_0 \) arbitrary assignment to \( \mathcal{E} \)
Solving by CEGAR Expansion Contd.

\[ \exists \mathcal{E} \forall \mathcal{U}. \phi \equiv \exists \mathcal{E}. \land_{\mu \in 2^{\mathcal{U}}} \phi[\mu] \]

Expand gradually instead: [Janota and Marques-Silva, 2011]

- Pick \( \tau_0 \) arbitrary assignment to \( \mathcal{E} \)
- \( \text{SAT}(\neg \phi[\tau_0]) = \mu_0 \) assignment to \( \mathcal{U} \)
\[ \exists \varepsilon \forall u. \phi \equiv \exists \varepsilon. \bigwedge_{\mu \in 2^u} \phi[\mu] \]

Expand **gradually** instead: [Janota and Marques-Silva, 2011]

- Pick \( \tau_0 \) arbitrary assignment to \( \varepsilon \)
- \( \text{SAT}(\neg \phi[\tau_0]) = \mu_0 \) assignment to \( U \)
- \( \text{SAT}(\phi[\mu_0]) = \tau_1 \) assignment to \( \varepsilon \)
Expand gradually instead: [Janota and Marques-Silva, 2011]

- Pick $\tau_0$ arbitrary assignment to $\mathcal{E}$
- $\text{SAT}(\neg \phi[\tau_0]) = \mu_0$ assignment to $\mathcal{U}$
- $\text{SAT}(\phi[\mu_0]) = \tau_1$ assignment to $\mathcal{E}$
- $\text{SAT}(\neg \phi[\tau_1]) = \mu_2$ assignment to $\mathcal{U}$
\[ \exists \mathcal{E} \forall \mathcal{U}. \phi \equiv \exists \mathcal{E}. \bigwedge_{\mu \in 2^\mathcal{U}} \phi[\mu] \]

Expand gradually instead: [Janota and Marques-Silva, 2011]

- Pick \( \tau_0 \) arbitrary assignment to \( \mathcal{E} \)
- \( \text{SAT}(\neg \phi[\tau_0]) = \mu_0 \) assignment to \( \mathcal{U} \)
- \( \text{SAT}(\phi[\mu_0]) = \tau_1 \) assignment to \( \mathcal{E} \)
- \( \text{SAT}(\neg \phi[\tau_1]) = \mu_2 \) assignment to \( \mathcal{U} \)
- \( \text{SAT}(\phi[\mu_0] \land \phi[\mu_1]) = \tau_2 \) assignment to \( \mathcal{E} \)
∃E ∀U. \phi \equiv \exists E. \land_{\mu \in 2^U} \phi[\mu]

Expand gradually instead: [Janota and Marques-Silva, 2011]

- Pick \(\tau_0\) arbitrary assignment to \(E\)
- \(\text{SAT}(\neg \phi[\tau_0]) = \mu_0\) assignment to \(U\)
- \(\text{SAT}(\phi[\mu_0]) = \tau_1\) assignment to \(E\)
- \(\text{SAT}(\neg \phi[\tau_1]) = \mu_2\) assignment to \(U\)
- \(\text{SAT}(\phi[\mu_0] \land \phi[\mu_1]) = \tau_2\) assignment to \(E\)
- After \(n\) iterations

\[\exists E. \land_{i \in 1..n} \phi[\tau_i]\]
Algorithm for $\exists \forall$. Generalize to arbitrary number of alternations using recursion. [Janota et al., 2012].

1. Function $\text{Solve}(\exists X \forall Y. \phi)$
2. $\alpha \leftarrow \text{true}$     // start with an empty abstraction
3. while true do
4.     $\tau \leftarrow \text{SAT}(\alpha)$     // find a candidate
5.     if $\tau = \bot$ then return $\bot$
6.     $\mu \leftarrow \text{Solve}(\neg \phi[X \leftarrow \tau])$     // find a countermove
7.     if $\mu = \bot$ then return $\tau$
8.     $\alpha \leftarrow \alpha \land \phi[Y \leftarrow \mu]$     // refine abstraction
Towards Machine Learning for Quantification
\( \exists x \ldots \forall y \ldots \phi \land y \)

Setting countermove \( y \leftarrow 0 \) yields false. Stop.
∃x ... ∀y .... ϕ ∧ y

Setting countermove $y \leftarrow 0$ yields false. Stop.

∃x ... ∀y .... x ∨ ϕ

Setting candidate $x \leftarrow 1$ yields true (impossible to falsify). Stop.
\exists x \forall y. x \Leftrightarrow y

1. \hspace{1cm} x \leftarrow 1
Careful Expansion: Bad Example

\exists x \forall y. x \Leftrightarrow y

1. \( x \leftarrow 1 \) candidate
2. \( \text{SAT}(\neg(1 \Leftrightarrow y)) \ldots y \leftarrow 0 \) countermove
$\exists x \forall y. \ x \iff y$

1. $x \leftarrow 1$
2. $\text{SAT}(\neg (1 \iff y)) \ldots y \leftarrow 0$
3. $\text{SAT}(x \iff 0) \ldots x \leftarrow 0$
\[\exists x \forall y. \ x \iff y\]

1. \(x \leftarrow 1\)  candidate
2. \(\text{SAT}(\lnot(1 \iff y)) \ldots y \leftarrow 0\)  countermove
3. \(\text{SAT}(x \iff 0) \ldots x \leftarrow 0\)  candidate
4. \(\text{SAT}(\lnot(0 \iff y)) \ldots y \leftarrow 1\)  countermove
Careful Expansion: Bad Example

\( \exists x \forall y. x \Leftrightarrow y \)

1. \( x \leftarrow 1 \)  \hspace{2cm} \text{candidate}
2. \( \text{SAT}(\neg(1 \Leftrightarrow y)) \ldots y \leftarrow 0 \)  \hspace{2cm} \text{countermove}
3. \( \text{SAT}(x \Leftrightarrow 0) \ldots x \leftarrow 0 \)  \hspace{2cm} \text{candidate}
4. \( \text{SAT}(\neg(0 \Leftrightarrow y)) \ldots y \leftarrow 1 \)  \hspace{2cm} \text{countermove}
5. \( \text{SAT}(x \Leftrightarrow 0 \land x \Leftrightarrow 1) \ldots \text{UNSAT} \)  \hspace{2cm} \text{Stop}
Careful Expansion: Ugly Example

\[ \exists x_1 x_2 \forall y_1 y_2. \; x_1 \iff y_1 \lor x_2 \iff y_2 \]

1. \( x_1, x_2 \leftarrow 0, 0 \)
\[ \exists x_1 x_2 \forall y_1 y_2. \ x_1 \iff y_1 \lor x_2 \iff y_2 \]

1. \( x_1, x_2 \leftarrow 0, 0 \)

2. \( \text{SAT}(\neg(0 \iff y_1 \lor \neg 0 \iff y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1 \)
Careful Expansion: Ugly Example

\[ \exists x_1 x_2 \forall y_1 y_2. \ x_1 \leftrightarrow y_1 \lor x_2 \leftrightarrow y_2 \]

1. \( x_1, x_2 \leftarrow 0, 0 \)
2. \( \text{SAT}(\neg (0 \leftrightarrow y_1 \lor \neg 0 \leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1 \)
3. \( \text{SAT}(x_1 \leftrightarrow 1 \lor x_2 \leftrightarrow 1) \ldots x_1, x_2 \leftarrow 0, 1 \)
Careful Expansion: Ugly Example

\[ \exists x_1 x_2 \forall y_1 y_2. \ x_1 \Leftrightarrow y_1 \lor x_2 \Leftrightarrow y_2 \]

1. \[ x_1, x_2 \leftarrow 0, 0 \]
2. \[ \text{SAT}(\neg (0 \Leftrightarrow y_1 \lor \neg 0 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1 \]
3. \[ \text{SAT}(x_1 \Leftrightarrow 1 \lor x_2 \Leftrightarrow 1) \ldots x_1, x_2 \leftarrow 0, 1 \]
4. \[ \text{SAT}(\neg (0 \Leftrightarrow y_1 \lor 1 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 0 \]
\[ \exists x_1 x_2 \forall y_1 y_2. \; x_1 \iff y_1 \lor x_2 \iff y_2 \]

1. \( x_1, x_2 \leftarrow 0, 0 \)
2. \( \text{SAT}(\neg(0 \iff y_1 \lor \neg0 \iff y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1 \)
3. \( \text{SAT}(x_1 \iff 1 \lor x_2 \iff 1) \ldots x_1, x_2 \leftarrow 0, 1 \)
4. \( \text{SAT}(\neg(0 \iff y_1 \lor 1 \iff y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 0 \)
5. \( \text{SAT}((x_1 \iff 1 \lor x_2 \iff 1) \land (x_1 \iff 1 \lor x_2 \iff 0)) \ldots \)
\[ \exists x_1 x_2 \forall y_1 y_2. \ x_1 \iff y_1 \lor x_2 \iff y_2 \]

1. \( x_1, x_2 \leftarrow 0, 0 \)
2. \( \text{SAT}(\neg (0 \iff y_1 \lor \neg 0 \iff y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1 \)
3. \( \text{SAT}(x_1 \iff 1 \lor x_2 \iff 1) \ldots x_1, x_2 \leftarrow 0, 1 \)
4. \( \text{SAT}(\neg (0 \iff y_1 \lor 1 \iff y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 0 \)
5. \( \text{SAT}((x_1 \iff 1 \lor x_2 \iff 1) \land (x_1 \iff 1 \lor x_2 \iff 0)) \ldots \)
6. \ldots \]
Learning in QBF
• CEGAR requires $2^n$ SAT calls for the formula

$$\exists x_1 \ldots x_n \forall y_1 \ldots y_n \cdot \bigvee_{i \in 1..n} x_i \leftrightarrow y_i$$
• CEGAR requires $2^n$ SAT calls for the formula

$$\exists x_1 \ldots x_n \forall y_1 \ldots y_n. \bigvee_{i \in 1..n} x_i \Leftrightarrow y_i$$

• BUT: We know that the formula is immediately false if we set $y_i \leftarrow \neg x_i$.

$$\left( \exists x_1 \ldots x_n \forall y_1 \ldots y_n. \bigvee_{i \in 1..n} x_i \Leftrightarrow \neg x_i \right) \equiv \left( \exists x_1 \ldots x_n. 0 \right)$$
• CEGAR requires $2^n$ SAT calls for the formula

$$\exists x_1 \ldots x_n \forall y_1 \ldots y_n. \bigvee_{i \in 1..n} x_i \leftrightarrow y_i$$

• **BUT:** We know that the formula is immediately false if we set $y_i \leftarrow \neg x_i$.

$$\left( \exists x_1 \ldots x_n \forall y_1 \ldots y_n. \bigvee_{i \in 1..n} x_i \leftrightarrow \neg x_i \right) \equiv \left( \exists x_1 \ldots x_n. \ 0 \right)$$

• **Idea:** instead of plugging in constants, plug in functions.
• CEGAR requires $2^n$ SAT calls for the formula

$$\exists x_1 \ldots x_n \forall y_1 \ldots y_n. \bigvee_{i \in 1..n} x_i \Leftrightarrow y_i$$

• **BUT:** We know that the formula is immediately false if we set $y_i \leftarrow \neg x_i$.

$$\left(\exists x_1 \ldots x_n \forall y_1 \ldots y_n. \bigvee_{i \in 1..n} x_i \Leftrightarrow \neg x_i\right) \equiv \left(\exists x_1 \ldots x_n. 0\right)$$

• **Idea:** instead of plugging in constants, plug in functions.

• **Where do we get the functions?**
[Janota, 2018]

1. Enumerate some number of candidate–countermove pairs.
[Janota, 2018]

1. Enumerate some number of candidate–countermove pairs.

2. Run a machine learning algorithm to learn a Boolean function for each variable in the inner quantifier.
1. Enumerate some number of candidate–countermove pairs.
2. Run a machine learning algorithm to learn a Boolean function for each variable in the inner quantifier.
3. Strengthen abstraction with the functions.
[Janota, 2018]

1. Enumerate some number of candidate–countermove pairs.
2. Run a machine learning algorithm to learn a Boolean function for each variable in the inner quantifier.
3. Strengthen abstraction with the functions.
4. Repeat.
Use Machine Learning

[Janota, 2018]

1. Enumerate some number of candidate–countermove pairs.
2. Run a machine learning algorithm to learn a Boolean function for each variable in the inner quantifier.
3. Strengthen abstraction with the functions.
4. Repeat.
5. Additional heuristic: If a learned function still works, keep it. “Don’t fix what ain’t broke.”
Machine Learning Example

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\ldots$</th>
<th>$x_n$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$\ldots$</th>
<th>$y_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$\ldots$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$\ldots$</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$\ldots$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\ldots$</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>$\ldots$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$\ldots$</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$\ldots$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$\ldots$</td>
<td>0</td>
</tr>
</tbody>
</table>
Machine Learning Example

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>...</th>
<th>$x_n$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>...</th>
<th>$y_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>

- After 2 steps: $y_1 \leftarrow \neg x_1$, $y_i \leftarrow 1$ for $i \in 2..n$. 

Eventually we learn the right functions.
Machine Learning Example

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>...</th>
<th>$x_n$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>...</th>
<th>$y_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>

• After 2 steps: $y_1 \leftarrow \neg x_1$, $y_i \leftarrow 1$ for $i \in 2..n$.
• $SAT(x_1 \leftrightarrow \neg x_1 \lor \lor_{i \in 2..n} x_2 \leftrightarrow 1)$
### Machine Learning Example

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>...</th>
<th>$x_n$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>...</th>
<th>$y_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>

- After 2 steps: $y_1 \leftarrow \neg x_1$, $y_i \leftarrow 1$ for $i \in 2..n$.
- $\text{SAT}(x_1 \iff \neg x_1 \lor \bigvee_{i \in 2..n} x_2 \iff 1)$
- After 4 steps: $y_1 \leftarrow \neg x_1$, $y_2 \leftarrow \neg x_2$ ...
Machine Learning Example

$$\begin{array}{ccccccc}
  x_1 & x_2 & \ldots & x_n & y_1 & y_2 & \ldots & y_n \\
  0 & 0 & \ldots & 0 & 1 & 1 & \ldots & 1 \\
  1 & 0 & \ldots & 0 & 0 & 1 & \ldots & 1 \\
  0 & 0 & \ldots & 1 & 1 & 1 & \ldots & 0 \\
  0 & 1 & \ldots & 1 & 1 & 0 & \ldots & 0 \\
\end{array}$$

- After 2 steps: $$y_1 \leftarrow \neg x_1, \ y_i \leftarrow 1 \text{ for } i \in 2..n.$$  
- $$\text{SAT}(x_1 \Leftrightarrow \neg x_1 \lor \bigvee_{i \in 2..n} x_2 \Leftrightarrow 1)$$
- After 4 steps: $$y_1 \leftarrow \neg x_1 \ y_2 \leftarrow \neg x_2 \ldots$$
- Eventually we learn the right functions.
Current Implementation

- Use CEGAR as before.
Current Implementation

- Use CEGAR as before.
- Recursion to generalize to multiple levels as before.
Current Implementation

- Use CEGAR as before.
- Recursion to generalize to multiple levels as before.
- Refinement as before.
Current Implementation

- Use CEGAR as before.
- Recursion to generalize to multiple levels as before.
- Refinement as before.
- Every $K$ refinements, learn new functions from last $K$ samples. Refine with them.
Current Implementation

- Use CEGAR as before.
- Recursion to generalize to multiple levels as before.
- Refinement as before.
- Every $K$ refinements, learn new functions from last $K$ samples. Refine with them.
- Learning using decision trees by ID3 algorithm.
Bernays–Schönfinkel ("Effectively Propositional Logic") — Finite Models
\( \forall X. \phi \)

- \( \phi \) has no further quantifiers and no functions (just predicates and constants)
∀X. ϕ

• ϕ has no further quantifiers and no functions (just predicates and constants)
• ϕ uses predicates $p_1, \ldots, p_m$ and constants $c_1, \ldots, c_n$. 
∀X. ϕ

- ϕ has no further quantifiers and no functions (just predicates and constants)
- ϕ uses predicates $p_1, \ldots, p_m$ and constants $c_1, \ldots, c_n$.
- **Finite model property**: formulas has a model iff it has a model of size $\leq n$. 
• \( \phi \) has no further quantifiers and no functions (just predicates and constants)

• \( \phi \) uses predicates \( p_1, \ldots, p_m \) and constants \( c_1, \ldots, c_n \).

• **Finite model property:** formulas has a model iff it has a model of size \( \leq n \).

• Therefore we can look for a model with the universe \( *_1, \ldots, *_{n'}, n' \leq n \).
\exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X. \phi

1. \alpha \leftarrow \text{true}
$\exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X. \phi$

$p_i$ predicates, $c_i$ constants, $X$ variables

1. $\alpha \leftarrow \text{true}$

2. Find interpretation for $\alpha$: $I \leftarrow \text{SAT}(\alpha)$
\[ \exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X. \phi \]

\( p_i \) predicates, \( c_i \) constants, \( X \) variables

1. \( \alpha \leftarrow \text{true} \)
2. Find interpretation for \( \alpha: \mathcal{I} \leftarrow \text{SAT}(\alpha) \)
3. Test interpretation: \( \mu \leftarrow \text{SAT}(\exists X. \neg \phi[I]) \)
\[ \exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X. \phi \]

- \( p_i \): predicates, \( c_i \): constants, \( X \): variables

1. \( \alpha \leftarrow \text{true} \)
2. Find interpretation for \( \alpha \): \( \mathcal{I} \leftarrow \text{SAT}(\alpha) \)
3. Test interpretation: \( \mu \leftarrow \text{SAT}(\exists X. \neg \phi[\mathcal{I}]) \)
4. If no counterexample, formula is true. STOP.
∃p_1 \ldots p_m ∃c_1 \ldots c_n ∀X. \phi

p_i \text{ predicates, } c_i \text{ constants, } X \text{ variables}

1. \alpha \leftarrow \text{true}
2. Find interpretation for \alpha: \mathcal{I} \leftarrow \text{SAT}(\alpha)
3. Test interpretation: \mu \leftarrow \text{SAT}(∃X. \neg \phi[\mathcal{I}])
4. If no counterexample, formula is true. STOP.
5. Strengthen abstraction: \alpha \leftarrow \alpha \land \phi[\mu/X]
\[ \exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X. \phi \]

\( p_i \) predicates, \( c_i \) constants, \( X \) variables

1. \( \alpha \leftarrow \text{true} \)
2. Find interpretation for \( \alpha \): \( I \leftarrow \text{SAT}(\alpha) \)
3. Test interpretation: \( \mu \leftarrow \text{SAT}(\exists X. \lnot \phi[I]) \)
4. If no counterexample, formula is true. \( \text{STOP.} \)
5. Strengthen abstraction: \( \alpha \leftarrow \alpha \land \phi[\mu/X] \)
6. \( \text{GOTO 2} \)
1. Consider some finite grounding:

\[ \exists p_1 \ldots p_m \exists c_1 \ldots c_n \land_{\mu \in \omega} \cdot \phi[\mu] \]

- \( p_i \) predicates, \( c_i \) constants,
1. Consider some finite grounding:
\[ \exists p_1 \ldots p_m \exists c_1 \ldots c_n \land_{\mu \in \omega} \cdot \phi[\mu] \]

\( p_i \) predicates, \( c_i \) constants,

2. Calculate interpretation by e.g. Ackermanization.
1. Consider some finite grounding:

$$\exists p_1 \ldots p_m \exists c_1 \ldots c_n \land_{\mu \in \omega} \phi[\mu]$$

$p_i$ predicates, $c_i$ constants,

2. Calculate interpretation by e.g. Ackermanization.

3. The interpretation only matters on the existing ground terms.
1. Consider some finite grounding:

\[ \exists p_1 \ldots p_m \exists c_1 \ldots c_n \bigwedge_{\mu \in \omega} \phi[\mu] \]

\( p_i \) predicates, \( c_i \) constants,

2. Calculate interpretation by e.g. Ackermanization.

3. The interpretation only matters on the existing ground terms.

4. Learn entire interpretation from observing values of existing terms.
1. $\forall X. \ p(X_1, \ldots, X_n) \Leftrightarrow (X_1 = t)$
1. $\forall X. p(X_1, \ldots, X_n) \iff (X_1 = t)$
2. Ground by $\{X_i \triangleq *_0\}$ and $\{X_1 \triangleq *_1, X_1 \triangleq *_0 \ldots X_n \triangleq *_0\}$:
1. $\forall X. p(X_1, \ldots, X_n) \Leftrightarrow (X_1 = t)$
2. Ground by $\{X_i \triangleq *_0\}$ and $\{X_1 \triangleq *_1, X_1 \triangleq *_0 \ldots X_n \triangleq *_0\}$:
3. $(p(*_0, \ldots, *_0) \Leftrightarrow *_0 = t) \land (p(*_1, \ldots, *_0) \Leftrightarrow *_1 = t)$
Learning in Finite Models’ CEGAR, Example

1. \( \forall X. p(X_1, \ldots, X_n) \iff (X_1 = t) \)
2. Ground by \( \{X_i \triangleq *_0\} \) and \( \{X_1 \triangleq *_1, X_1 \triangleq *_0 \ldots X_n \triangleq *_0\} \):
3. \( (p(*_0, \ldots, *_0) \iff *_0 = t) \land (p(*_1, \ldots, *_0) \iff *_1 = t) \)
4. Partial interpretation:
   \[ t \triangleq *_1, p(*_0 \ldots, *_0) \triangleq \text{False}, p(*_1 \ldots, *_0) \triangleq \text{True} \]
Learning in Finite Models’ CEGAR, Example

1. \( \forall X. p(X_1, \ldots, X_n) \Leftrightarrow (X_1 = t) \)

2. Ground by \( \{ X_i \triangleq *_0 \} \) and \( \{ X_1 \triangleq *_1, X_1 \triangleq *_0, \ldots, X_n \triangleq *_0 \} \):

3. \((p(*_0, \ldots, *_0) \Leftrightarrow *_0 = t) \land (p(*_1, \ldots, *_0) \Leftrightarrow *_1 = t)\)

4. Partial interpretation:
   \( t \triangleq *_1, p(*_0 \ldots, *_0) \triangleq \text{False}, p(*_1 \ldots, *_0) \triangleq \text{True} \)

5. Learn: \( t \triangleq *_1, p(X_1, \ldots, X_n) \triangleq (X_1 = *_1) \),
Preliminary Results

Towards Machine Learning for Quantification
Preliminary Results

Towards Machine Learning for Quantification
Preliminary Results (Hard) - more then 1 sec

![Graph showing CPU time (s) vs. instances for different methods: cegar+learn, cegar, expand]
Learn vs. CEGAR, Iterations — Only True

Janota
Towards Machine Learning for Quantification
Summary and Future

- Observing a formula while solving, learn from that.
Summary and Future

- Observing a formula while solving, learn from that.
- Learning objects in the considered theory. (rather than strategies, etc.)
Summary and Future

- Observing a formula while solving, learn from that.
- Learning objects in the considered theory. (rather than strategies, etc.)
- Learning from Booleans:
  
  For \( \exists B^n \forall B^m \ldots \), learning \( B^n \rightarrow B \)
Summary and Future

• Observing a formula while solving, learn from that.
• Learning objects in the considered theory. (rather than strategies, etc.)
• Learning from Booleans:
  For \( \exists B^n \forall B^m \ldots \), learning \( B^n \rightarrow B \)
• Learning interpretations in finite models from partial interpretations:
  For \( \exists (D_1 \times \cdots \times D_k \rightarrow B) \forall F_1 \times \cdots \times F_l \ldots \),
  learning \( D_1 \times \cdots \times D_k \rightarrow B \)
Summary and Future

- Observing a formula while solving, learn from that.
- Learning objects in the considered theory. (rather than strategies, etc.)
- Learning from Booleans:
  \[ \exists B^n \forall B^m \ldots, \text{learning } B^n \rightarrow B \]
- Learning interpretations in finite models from partial interpretations:
  \[ \exists (D_1 \times \cdots \times D_k \mapsto B) \forall F_1 \times \cdots \times F_l \ldots, \]
  learning \( D_1 \times \cdots \times D_k \rightarrow B \)
- How can we learn strategies based on functions?

Janota
Towards Machine Learning for Quantification
Summary and Future

- Observing a formula while solving, learn from that.
- Learning objects in the considered theory. (rather than strategies, etc.)
- Learning from Booleans:
  \[
  \text{For } \exists B^n \land B^m \ldots, \text{ learning } B^n \to B
  \]
- Learning interpretations in finite models from partial interpretations:
  \[
  \text{For } \exists (D_1 \times \cdots \times D_k \mapsto B) \land F_1 \times \cdots \times F_l \ldots, \text{ learning } D_1 \times \cdots \times D_k \to B
  \]
- How can we learn strategies based on functions?
- Infinite domains?
Summary and Future

• Observing a formula while solving, learn from that.
• Learning objects in the considered theory. (rather than strategies, etc.)
• Learning from Booleans:
  \[ \text{For } \exists B^n \forall B^m \ldots, \text{ learning } B^n \rightarrow B \]
• Learning interpretations in finite models from partial interpretations:
  \[ \text{For } \exists (D_1 \times \cdots \times D_k \mapsto B) \forall F_1 \times \cdots \times F_l \ldots, \]
  learning \( D_1 \times \cdots \times D_k \rightarrow B \)
• How can we learn strategies based on functions?
• Infinite domains?
• Learning in the presence of theories?
Thank You for Your Attention!

Questions?
