Towards Machine Learning for Quantification

Mikoláš Janota

AITP, 28 March 2018

IST/INESC-ID,
University of Lisbon, Portugal

Outline

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: $\{x = 1, y = 0\} \models (x \vee y) \wedge (x \vee \neg y)$

SAT and QBF

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

- SAT — for a Boolean formula, determine if it is satisfiable
- Example: $\{x = 1, y = 0\} \models (x \vee y) \wedge (x \vee \neg y)$
- QBF — for a *Quantified* Boolean formula
- Example: $\forall x \exists y.\ (x \leftrightarrow y)$

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

  Example:

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

  Example:

  (1) $\forall x \exists y.\ (x \leftrightarrow y)$

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

  Example:

  (1) $\forall x \exists y.\ (x \leftrightarrow y)$

  (2) $\forall x.\ (x \leftrightarrow 0) \vee (x \leftrightarrow 1)$

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

  Example:

  (1) $\forall x \exists y. \ (x \leftrightarrow y)$

  (2) $\forall x. \ (x \leftrightarrow 0) \vee (x \leftrightarrow 1)$

  (3) $((0 \leftrightarrow 0) \vee (0 \leftrightarrow 1)) \wedge ((1 \leftrightarrow 0) \vee (1 \leftrightarrow 1))$

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

  Example:

  (1) $\forall x \exists y.\ (x \leftrightarrow y)$
  (2) $\forall x.\ (x \leftrightarrow 0) \vee (x \leftrightarrow 1)$
  (3) $((0 \leftrightarrow 0) \vee (0 \leftrightarrow 1)) \wedge ((1 \leftrightarrow 0) \vee (1 \leftrightarrow 1))$
  (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,

- 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:

$$\text{is-true}(t) \wedge \neg\text{is-true}(f) \wedge$$
$$(\forall X_u.\ \text{is-true}(X_u) \leftrightarrow p_e(X_u))$$

- 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:

$$\text{is-true}(t) \wedge \neg\text{is-true}(f) \wedge$$
$$(\forall X_u.\ \text{is-true}(X_u) \leftrightarrow p_e(X_u))$$

- Alternatively, use equality:

$$t \neq f \wedge (\forall X_u.\ (X_u = t) \leftrightarrow p_e(X_u))$$

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

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

- In this talk we consider prenex form:
  Quantifier-prefix. Matrix
  Example $\forall u_1 u_2 \exists e_1 e_2 .\, (\neg u_1 \vee e_1) \wedge (u_2 \vee \neg e_2)$
- A QBF represents a two-player game between $\forall$ and $\exists$.

- In this talk we consider prenex form:
  Quantifier-prefix. *Matrix*
  Example $\forall u_1 u_2 \exists e_1 e_2 .\, (\neg u_1 \vee e_1) \wedge (u_2 \vee \neg e_2)$
- A QBF represents a two-player game between $\forall$ and $\exists$.
- $\forall$ wins a game if the matrix becomes false.

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

TÉCNICO LISBOA

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

- In this talk we consider prenex form:
  Quantifier-prefix. Matrix
  Example $\forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \vee e_1) \wedge (u_2 \vee \neg e_2)$
- 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 $e \triangleq u$.

Solving QBF

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

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

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

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

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

Observe:

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

for some $\omega \subseteq 2^{\mathcal{U}}$

What is a good ω?

Solving by CEGAR Expansion Contd.

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

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

- Pick τ_0 arbitrary assignment to $\mathcal{E}$

TÉCNICO LISBOA

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

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

- Pick τ_0 arbitrary assignment to $\mathcal{E}$
- $\mathsf{SAT}(\neg\phi[\tau_0]) = \mu_0$ 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 τ_0 arbitrary assignment to $\mathcal{E}$
- $\mathsf{SAT}(\neg\phi[\tau_0]) = \mu_0$ assignment to $\mathcal{U}$
- $\mathsf{SAT}(\phi[\mu_0]) = \tau_1$ assignment to $\mathcal{E}$

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

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

- Pick τ_0 arbitrary assignment to $\mathcal{E}$
- $\mathsf{SAT}(\neg\phi[\tau_0]) = \mu_0$ assignment to $\mathcal{U}$
- $\mathsf{SAT}(\phi[\mu_0]) = \tau_1$ assignment to $\mathcal{E}$
- $\mathsf{SAT}(\neg\phi[\tau_1]) = \mu_2$ assignment to $\mathcal{U}$

Solving by CEGAR Expansion Contd.

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

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

- Pick τ_0 arbitrary assignment to $\mathcal{E}$
- $\mathsf{SAT}(\neg\phi[\tau_0]) = \mu_0$ assignment to $\mathcal{U}$
- $\mathsf{SAT}(\phi[\mu_0]) = \tau_1$ assignment to $\mathcal{E}$
- $\mathsf{SAT}(\neg\phi[\tau_1]) = \mu_2$ assignment to $\mathcal{U}$
- $\mathsf{SAT}(\phi[\mu_0] \wedge \phi[\mu_1]) = \tau_2$ assignment to $\mathcal{E}$

Solving by CEGAR Expansion Contd.

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

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

- Pick τ_0 arbitrary assignment to $\mathcal{E}$
- $\mathsf{SAT}(\neg\phi[\tau_0]) = \mu_0$ assignment to $\mathcal{U}$
- $\mathsf{SAT}(\phi[\mu_0]) = \tau_1$ assignment to $\mathcal{E}$
- $\mathsf{SAT}(\neg\phi[\tau_1]) = \mu_2$ assignment to $\mathcal{U}$
- $\mathsf{SAT}(\phi[\mu_0] \wedge \phi[\mu_1]) = \tau_2$ assignment to $\mathcal{E}$
- After n iterations

$$\exists \mathcal{E}.\ \bigwedge_{i \in 1..n} \phi[\tau_i]$$

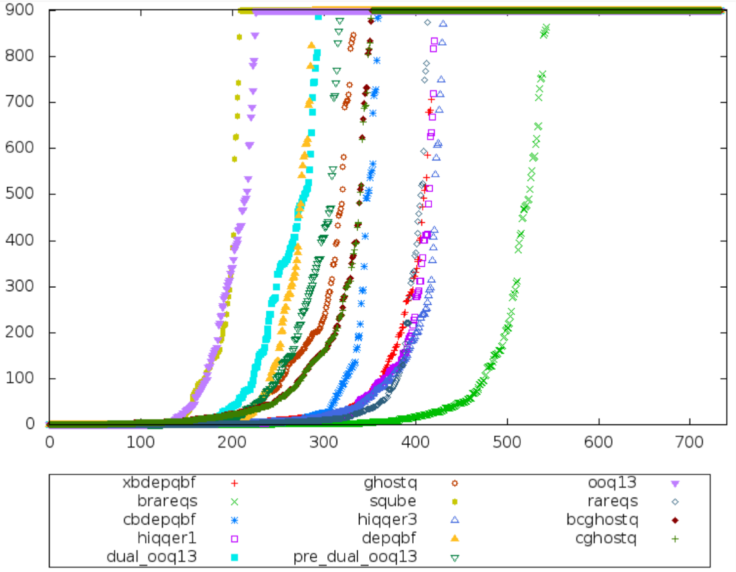
Abstraction-Based Algorithm for a Winning Move

Algorithm for $\exists\forall$. Generalize to arbitrary number of alternations using recursion. [Janota et al., 2012].

Function `Solve`($\exists X \forall Y.\, \phi$)

$\alpha \leftarrow$ true // start with an empty abstraction
while true **do**
  $\tau \leftarrow \mathsf{SAT}(\alpha)$ // find a candidate
  if $\tau = \bot$ **then return** $\bot$
  $\mu \leftarrow$ `Solve`($\neg\phi[X \leftarrow \tau]$) // find a countermove
  if $\mu = \bot$ **then return** τ
  $\alpha \leftarrow \alpha \wedge \phi[Y \leftarrow \mu]$ // refine abstraction

Results, QBF-Gallery ’14, Application Track



Careful Expansion: Good Example

$$\exists x \ldots \forall y \ldots . \phi \wedge y$$

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

$\exists x \ldots \forall y \ldots .\ \phi \wedge y$

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

$\exists x \ldots \forall y \ldots .\ x \vee \phi$

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

Careful Expansion: Bad Example

$\exists x \forall y.\ x \Leftrightarrow y$

1. $x \leftarrow 1$ candidate

Careful Expansion: Bad Example

$\exists x \forall y.\ x \Leftrightarrow y$

1. $x \leftarrow 1$ candidate
2. $\mathsf{SAT}(\neg(1 \Leftrightarrow y)) \ldots y \leftarrow 0$ countermove

Careful Expansion: Bad Example

$\exists x \forall y.\ x \Leftrightarrow y$

1. $x \leftarrow 1$ candidate
2. $\mathsf{SAT}(\neg(1 \Leftrightarrow y)) \ldots y \leftarrow 0$ countermove
3. $\mathsf{SAT}(x \Leftrightarrow 0) \ldots x \leftarrow 0$ candidate

TÉCNICO LISBOA

$$\exists x \forall y.\ x \Leftrightarrow y$$

1. $x \leftarrow 1$ candidate
2. $\mathsf{SAT}(\neg(1 \Leftrightarrow y)) \ldots y \leftarrow 0$ countermove
3. $\mathsf{SAT}(x \Leftrightarrow 0) \ldots x \leftarrow 0$ candidate
4. $\mathsf{SAT}(\neg(0 \Leftrightarrow y)) \ldots y \leftarrow 1$ countermove

TÉCNICO LISBOA

$\exists x \forall y.\ x \Leftrightarrow y$

1. $x \leftarrow 1$ candidate
2. $\mathsf{SAT}(\neg(1 \Leftrightarrow y)) \ldots y \leftarrow 0$ countermove
3. $\mathsf{SAT}(x \Leftrightarrow 0) \ldots x \leftarrow 0$ candidate
4. $\mathsf{SAT}(\neg(0 \Leftrightarrow y)) \ldots y \leftarrow 1$ countermove
5. $\mathsf{SAT}(x \Leftrightarrow 0 \wedge x \Leftrightarrow 1) \ldots$ UNSAT Stop

Careful Expansion: Ugly Example

$$\exists x_1 x_2 \forall y_1 y_2.\ x_1 \Leftrightarrow y_1 \vee x_2 \Leftrightarrow y_2$$

1. $x_1, x_2 \leftarrow 0, 0$

Careful Expansion: Ugly Example

$$\exists x_1 x_2 \forall y_1 y_2.\ x_1 \Leftrightarrow y_1 \vee x_2 \Leftrightarrow y_2$$

1. $x_1, x_2 \leftarrow 0, 0$
2. $\mathsf{SAT}(\neg(0 \Leftrightarrow y_1 \vee \neg 0 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1$

Careful Expansion: Ugly Example

$$\exists x_1 x_2 \forall y_1 y_2.\ x_1 \Leftrightarrow y_1 \vee x_2 \Leftrightarrow y_2$$

1. $x_1, x_2 \leftarrow 0, 0$
2. $\mathsf{SAT}(\neg(0 \Leftrightarrow y_1 \vee \neg 0 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1$
3. $\mathsf{SAT}(x_1 \Leftrightarrow 1 \vee x_2 \Leftrightarrow 1) \ldots x_1, x_2 \leftarrow 0, 1$

$$\exists x_1 x_2 \forall y_1 y_2.\ x_1 \Leftrightarrow y_1 \vee x_2 \Leftrightarrow y_2$$

1. $x_1, x_2 \leftarrow 0, 0$
2. $\mathsf{SAT}(\neg(0 \Leftrightarrow y_1 \vee \neg 0 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1$
3. $\mathsf{SAT}(x_1 \Leftrightarrow 1 \vee x_2 \Leftrightarrow 1) \ldots x_1, x_2 \leftarrow 0, 1$
4. $\mathsf{SAT}(\neg(0 \Leftrightarrow y_1 \vee 1 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 0$

Careful Expansion: Ugly Example

$$\exists x_1 x_2 \forall y_1 y_2.\ x_1 \Leftrightarrow y_1 \vee x_2 \Leftrightarrow y_2$$

1. $x_1, x_2 \leftarrow 0, 0$
2. $\mathsf{SAT}(\neg(0 \Leftrightarrow y_1 \vee \neg 0 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1$
3. $\mathsf{SAT}(x_1 \Leftrightarrow 1 \vee x_2 \Leftrightarrow 1) \ldots x_1, x_2 \leftarrow 0, 1$
4. $\mathsf{SAT}(\neg(0 \Leftrightarrow y_1 \vee 1 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 0$
5. $\mathsf{SAT}\big((x_1 \Leftrightarrow 1 \vee x_2 \Leftrightarrow 1) \wedge (x_1 \Leftrightarrow 1 \vee x_2 \Leftrightarrow 0)\big) \ldots$

$$\exists x_1 x_2 \forall y_1 y_2 .\ x_1 \Leftrightarrow y_1 \vee x_2 \Leftrightarrow y_2$$

1. $x_1, x_2 \leftarrow 0, 0$
2. $\mathsf{SAT}(\neg(0 \Leftrightarrow y_1 \vee \neg 0 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1$
3. $\mathsf{SAT}(x_1 \Leftrightarrow 1 \vee x_2 \Leftrightarrow 1) \ldots x_1, x_2 \leftarrow 0, 1$
4. $\mathsf{SAT}(\neg(0 \Leftrightarrow y_1 \vee 1 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 0$
5. $\mathsf{SAT}\big((x_1 \Leftrightarrow 1 \vee x_2 \Leftrightarrow 1) \wedge (x_1 \Leftrightarrow 1 \vee x_2 \Leftrightarrow 0)\big) \ldots$
6. $\ldots$

Learning in QBF

Issue

- CEGAR requires 2^n SAT calls for the formula

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

Issue

- 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 $y_i \leftarrow \neg x_i$.

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

Issue

- 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.

Issue

- 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?**

Use Machine Learning

[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.

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.

[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.”

x_1	x_2	...	x_n	y_1	y_2	...	y_n
0	0	...	0	1	1	...	1
1	0	...	0	0	1	...	1
0	0	...	1	1	1	...	0
0	1	...	1	1	0	...	0

x_1	x_2	...	x_n	y_1	y_2	...	y_n
0	0	...	0	1	1	...	1
1	0	...	0	0	1	...	1
0	0	...	1	1	1	...	0
0	1	...	1	1	0	...	0

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

TÉCNICO LISBOA

x_1	x_2	$\dots$	x_n	y_1	y_2	$\dots$	y_n
0	0	$\dots$	0	1	1	$\dots$	1
1	0	$\dots$	0	0	1	$\dots$	1
0	0	$\dots$	1	1	1	$\dots$	0
0	1	$\dots$	1	1	0	$\dots$	0

- After 2 steps: $y_1 \leftarrow \neg x_1$, $y_i \leftarrow 1$ for $i \in 2..n$.
- $SAT(x_1 \Leftrightarrow \neg x_1 \vee \bigvee_{i\in 2..n} x_2 \Leftrightarrow 1)$

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

- After 2 steps: $y_1 \leftarrow \neg x_1$, $y_i \leftarrow 1$ for $i \in 2..n$.
- $SAT(x_1 \Leftrightarrow \neg x_1 \vee \bigvee_{i \in 2..n} x_2 \Leftrightarrow 1)$
- After 4 steps: $y_1 \leftarrow \neg x_1\ y_2 \leftarrow \neg x_2 \ldots$

TÉCNICO LISBOA

x_1	x_2	...	x_n	y_1	y_2	...	y_n
0	0	...	0	1	1	...	1
1	0	...	0	0	1	...	1
0	0	...	1	1	1	...	0
0	1	...	1	1	0	...	0

- After 2 steps: $y_1 \leftarrow \neg x_1$, $y_i \leftarrow 1$ for $i \in 2..n$.
- $SAT(x_1 \Leftrightarrow \neg x_1 \vee \bigvee_{i\in 2..n} x_2 \Leftrightarrow 1)$
- 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.

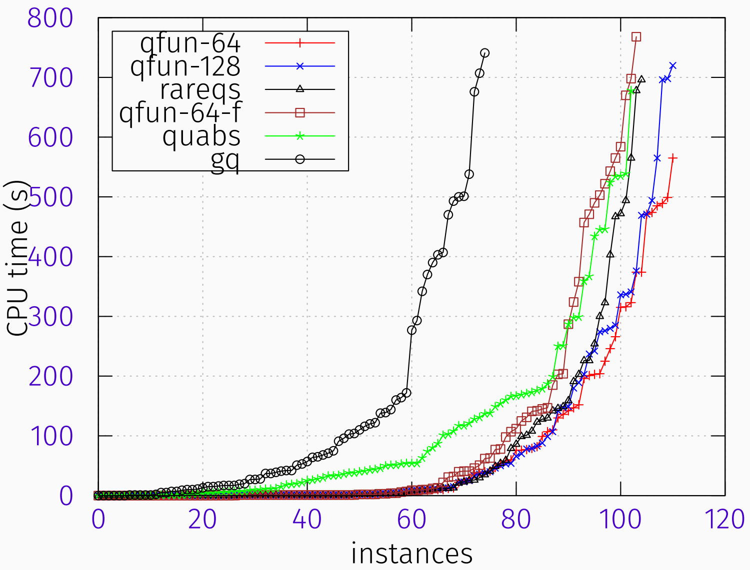
Current Implementation

- 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.

Current Implementation: Experiments



Bernays–Schönfinkel (“Effectively Propositional Logic”) — Finite Models

Bernays–Schönfinkel (EPR)

$$\forall X.\ \phi$$

- ϕ has no further quantifiers and no functions (just predicates and constants)

$$\forall X.\ \phi$$

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

$$\forall X.\ \phi$$

- 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 $\leq n$.

$$\forall X.\ \phi$$

- 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 $\leq n$.
- Therefore we can look for a model with the universe $*_1, \ldots, *_{n'}$, $n' \leq n$.

$$\exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X.\ \phi$$

p_i predicates, c_i constants, X variables

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

$$\exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X.\ \phi$$

p_i predicates, c_i constants, X variables

1. $\alpha \leftarrow$ `true`
2. Find interpretation for α: $\mathcal{I} \leftarrow \mathtt{SAT}(\alpha)$

$$\exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X.\ \phi$$

p_i predicates, c_i constants, X variables

1. $\alpha \leftarrow \texttt{true}$
2. Find interpretation for α: $\mathcal{I} \leftarrow \mathsf{SAT}(\alpha)$
3. Test interpretation: $\mu \leftarrow \mathsf{SAT}(\exists X.\ \neg\phi[\mathcal{I}])$

$$\exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X.\ \phi$$

p_i predicates, c_i constants, X variables

1. $\alpha \leftarrow \texttt{true}$
2. Find interpretation for α: $\mathcal{I} \leftarrow \mathsf{SAT}(\alpha)$
3. Test interpretation: $\mu \leftarrow \mathsf{SAT}(\exists X.\ \neg\phi[\mathcal{I}])$
4. If no counterexample, formula is true. STOP.

$$\exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X.\ \phi$$

p_i predicates, c_i constants, X variables

1. $\alpha \leftarrow$ `true`
2. Find interpretation for α: $\mathcal{I} \leftarrow \mathsf{SAT}(\alpha)$
3. Test interpretation: $\mu \leftarrow \mathsf{SAT}(\exists X.\ \neg\phi[\mathcal{I}])$
4. If no counterexample, formula is true. STOP.
5. Strengthen abstraction: $\alpha \leftarrow \alpha \wedge \phi[\mu/X]$

$\exists p_1 \ldots p_m \exists c_1 \ldots c_n \forall X.\ \phi$

p_i predicates, c_i constants, X variables

1. $\alpha \leftarrow \texttt{true}$
2. Find interpretation for α: $\mathcal{I} \leftarrow \mathsf{SAT}(\alpha)$
3. Test interpretation: $\mu \leftarrow \mathsf{SAT}(\exists X.\ \neg\phi[\mathcal{I}])$
4. If no counterexample, formula is true. STOP.
5. Strengthen abstraction: $\alpha \leftarrow \alpha \wedge \phi[\mu/X]$
6. GOTO 2

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,

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.

Learning in Finite Models' CEGAR

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.

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.

Learning in Finite Models' CEGAR, Example

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

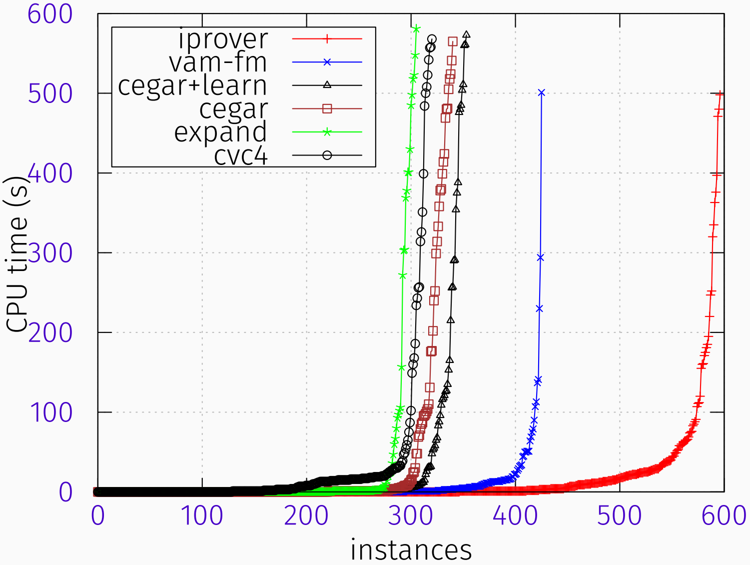
1. $\forall X.\ p(X_1, \ldots, X_n) \Leftrightarrow (X_1 = t)$
2. Ground by $\{X_i \triangleq *_0\}$ and $\{X_1 \triangleq *_1, X_1 \triangleq *_0 \ldots X_n \triangleq *_0\}$:

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. $(p(*_0, \ldots, *_0) \Leftrightarrow *_0 = t) \wedge (p(*_1, \ldots, *_0) \Leftrightarrow *_1 = t)$

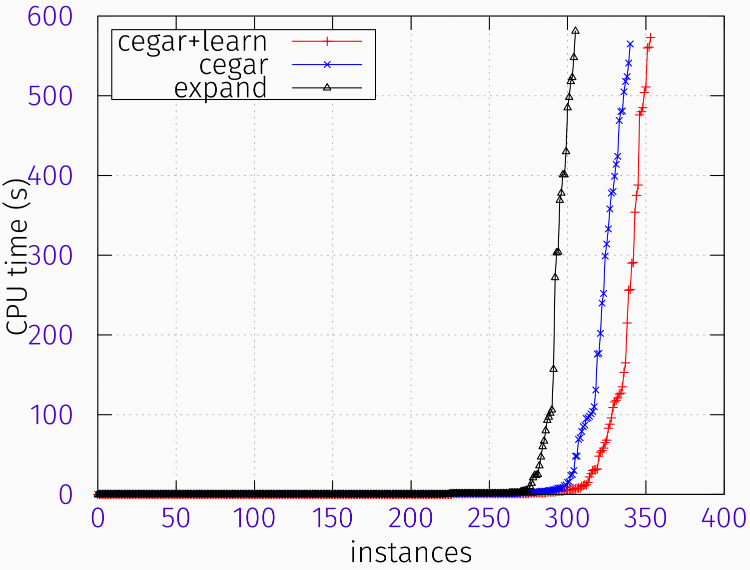
1. $\forall X.\ p(X_1, \ldots, X_n) \Leftrightarrow (X_1 = t)$
2. Ground by $\{X_i \triangleq *_0\}$ and $\{X_1 \triangleq *_1, X_1 \triangleq *_0 \ldots X_n \triangleq *_0\}$:
3. $(p(*_0, \ldots, *_0) \Leftrightarrow *_0 = t) \wedge (p(*_1, \ldots, *_0) \Leftrightarrow *_1 = t)$
4. Partial interpretation:
   $t \triangleq *_1, p(*_0 \ldots, *_0) \triangleq \text{False}, p(*_1 \ldots, *_0) \triangleq \text{True}$

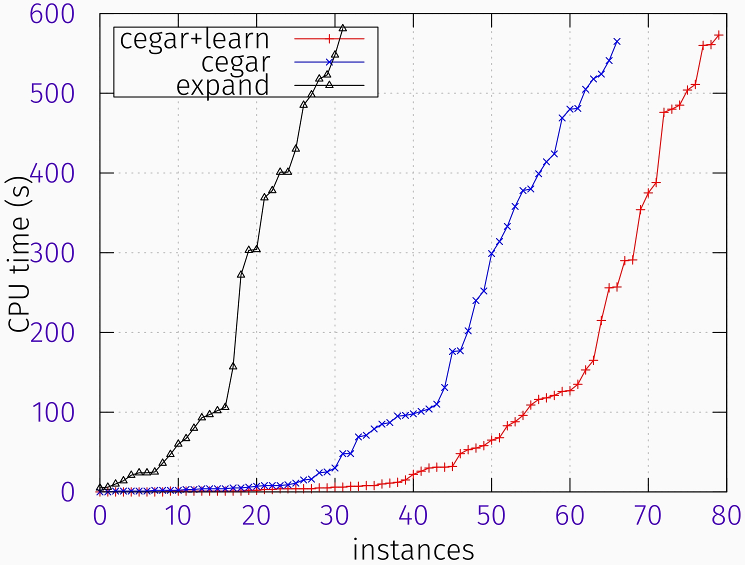
Learning in Finite Models' CEGAR, Example

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. $(p(*_0, \ldots, *_0) \Leftrightarrow *_0 = t) \wedge (p(*_1, \ldots, *_0) \Leftrightarrow *_1 = t)$
4. Partial interpretation:
   $t \triangleq *_1, p(*_0 \ldots, *_0) \triangleq \text{False}, p(*_1 \ldots, *_0) \triangleq \text{True}$
5. Learn: $t \triangleq *_1$, $p(X_1, \ldots, X_n) \triangleq (X_1 = *_1)$,


iprover
vam-fm
cegar+learn
cegar
expand
cvc4
CPU time (s)
0
100
200
300
400
500
600
0
100
200
300
400
500
600
instances

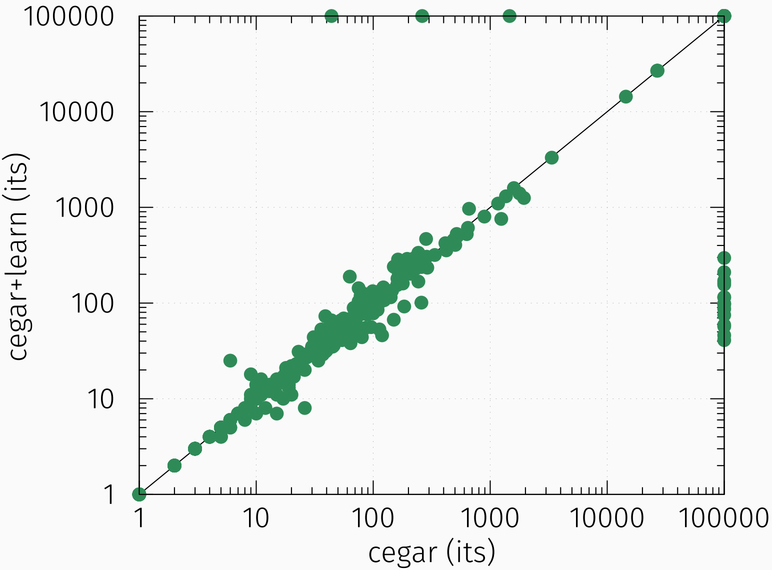
Preliminary Results



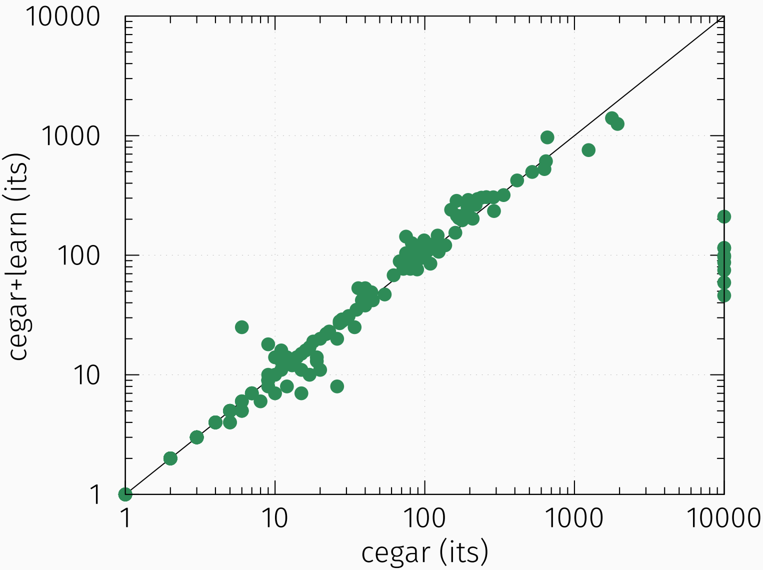


cegar+learn
cegar
expand
CPU time (s)
instances
0
100
200
300
400
500
600
0
10
20
30
40
50
60
70
80

Learn vs. CEGAR, Iterations



Learn vs. CEGAR, Iterations — Only True



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:

$$\text{For } \ldots \exists \mathbb{B}^n \forall \mathbb{B}^m \ldots, \text{ learning } \mathbb{B}^n \to \mathbb{B}$$

TÉCNICO LISBOA

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

  For $\ldots \exists \mathbb{B}^n \forall \mathbb{B}^m \ldots$, learning $\mathbb{B}^n \to \mathbb{B}$
- Learning interpretations in finite models from partial interpretations:
  For $\exists (D_1 \times \cdots \times D_k \mapsto \mathbb{B}) \forall F_1 \times \cdots \times F_l \ldots$,
  learning $D_1 \times \cdots \times D_k \to \mathbb{B}$

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 $\dots \exists \mathbb{B}^n \forall \mathbb{B}^m \dots$, learning $\mathbb{B}^n \to \mathbb{B}$
- Learning interpretations in finite models from partial interpretations:
  For $\exists (D_1 \times \cdots \times D_k \mapsto \mathbb{B}) \forall F_1 \times \cdots \times F_l \dots$,
  learning $D_1 \times \cdots \times D_k \to \mathbb{B}$
- How can we learn strategies based on functions?

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 $\dots \exists \mathbb{B}^n \forall \mathbb{B}^m \dots$, learning $\mathbb{B}^n \to \mathbb{B}$
- Learning interpretations in finite models from partial interpretations:

  For $\exists (D_1 \times \cdots \times D_k \mapsto \mathbb{B}) \forall F_1 \times \cdots \times F_l \dots$,
  learning $D_1 \times \cdots \times D_k \to \mathbb{B}$
- 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:

$$\text{For } \ldots \exists \mathbb{B}^n \forall \mathbb{B}^m \ldots, \text{ learning } \mathbb{B}^n \to \mathbb{B}$$

- Learning interpretations in finite models from partial interpretations:
  For $\exists(D_1 \times \cdots \times D_k \mapsto \mathbb{B})\forall F_1 \times \cdots \times F_l \ldots,$
  learning $D_1 \times \cdots \times D_k \to \mathbb{B}$
- How can we learn strategies based on functions?
- Infinite domains?
- Learning in the presence of theories?

Thank You for Your Attention!

Questions?

Janota, M. (2018).
Towards generalization in QBF solving via machine learning.
In *AAAI Conference on Artificial Intelligence*.

Janota, M., Klieber, W., Marques-Silva, J., and Clarke, E. M. (2012).
Solving QBF with counterexample guided refinement.
In *SAT*, pages 114–128.

Janota, M. and Marques-Silva, J. (2011).
Abstraction-based algorithm for 2QBF.
In *SAT*.