Towards Machine Learning for SMT

Mikoláš Janota

MLR, 30 April 2020

Outline

Intro: QBF, Expansion, Games, Careful expansion

Solving QBF

Learning in QBF

Towards SMT

Intro: QBF, Expansion, Games, Careful expansion

SAT and QBF

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

SAT and QBF

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

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

SAT and QBF

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

SMT Satisfiability Modulo Theories

- reasoning in first order logic

SMT Satisfiability Modulo Theories

- reasoning in first order logic
- under a given theory, e.g. *linear arithmetic without quantifiers*

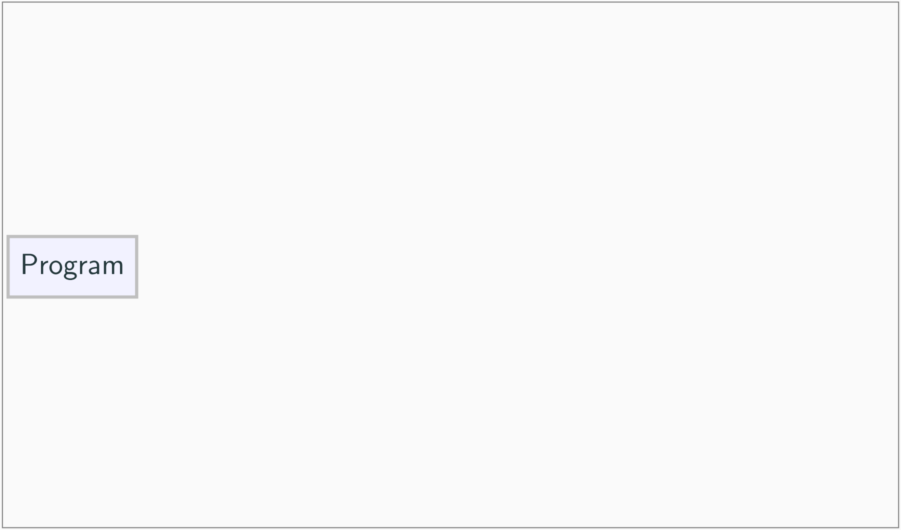
SMT Satisfiability Modulo Theories

- reasoning in first order logic
- under a given theory, e.g. *linear arithmetic without quantifiers*
- focus: software verification, debugging, ...

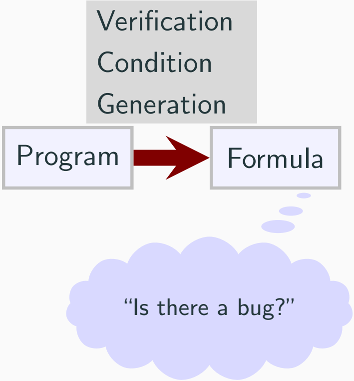
SMT Satisfiability Modulo Theories

- reasoning in first order logic
- under a given theory, e.g. *linear arithmetic without quantifiers*
- focus: software verification, debugging, ...
- formulas may be huge, solvers are not based on saturation but on SAT technology

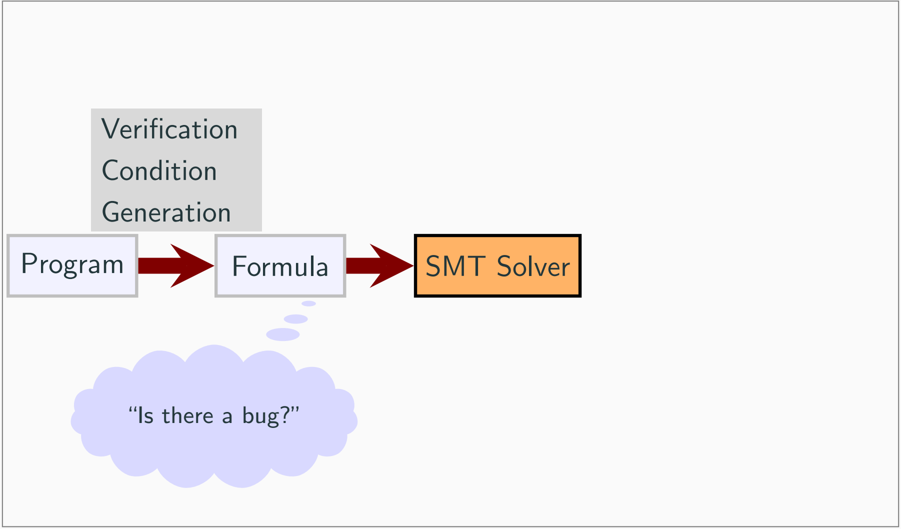
How is SMT Used in SW Verification



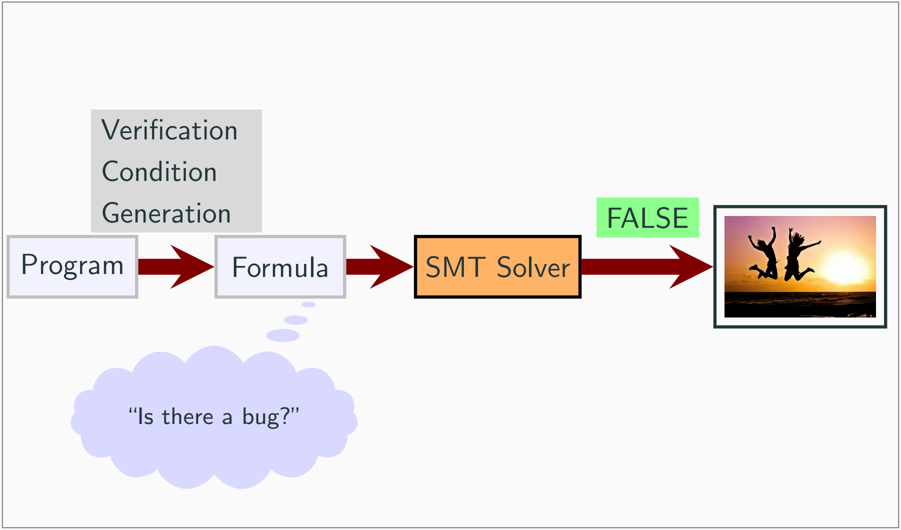
How is SMT Used in SW Verification



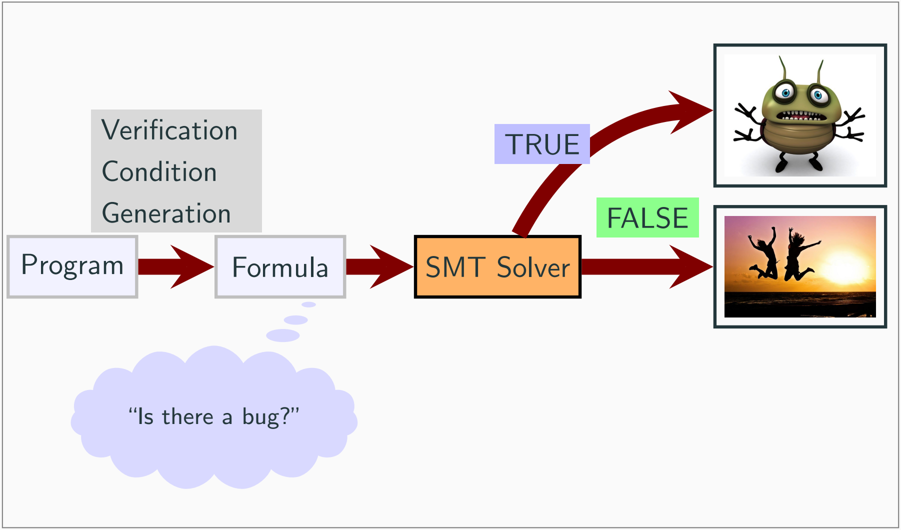
How is SMT Used in SW Verification



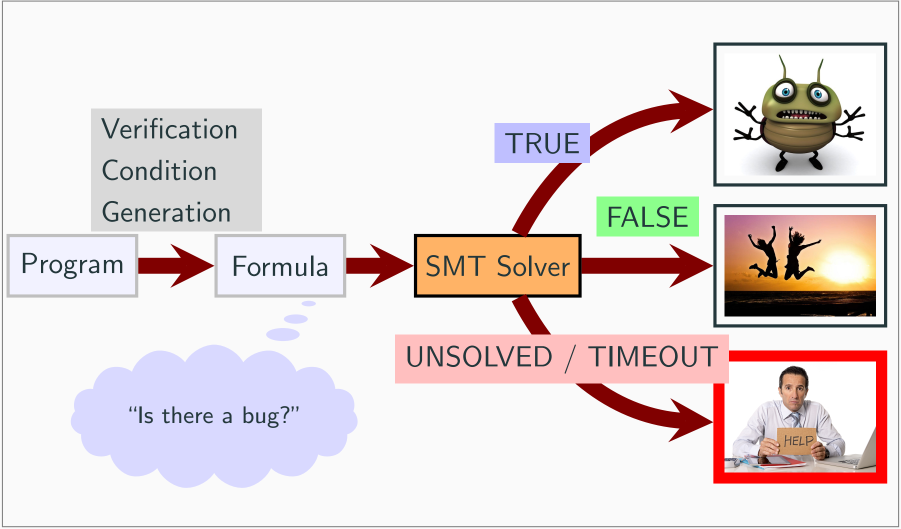
How is SMT Used in SW Verification



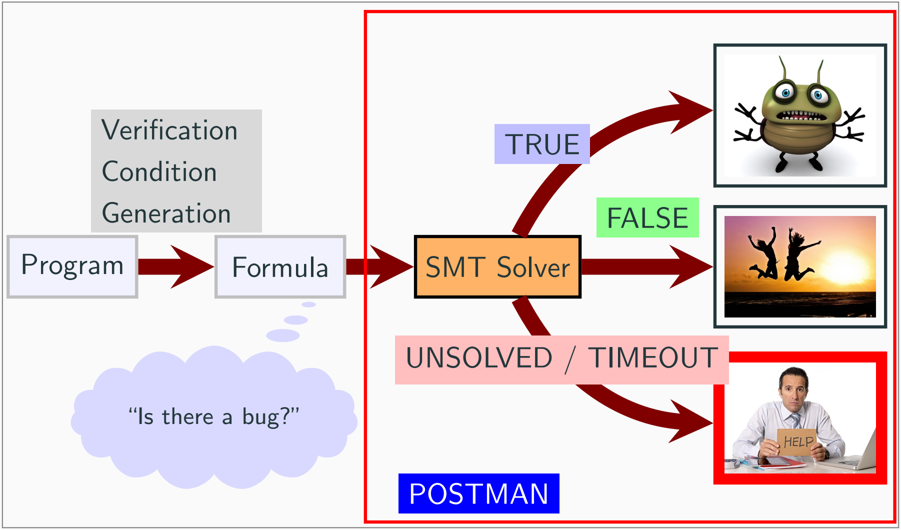
How is SMT Used in SW Verification



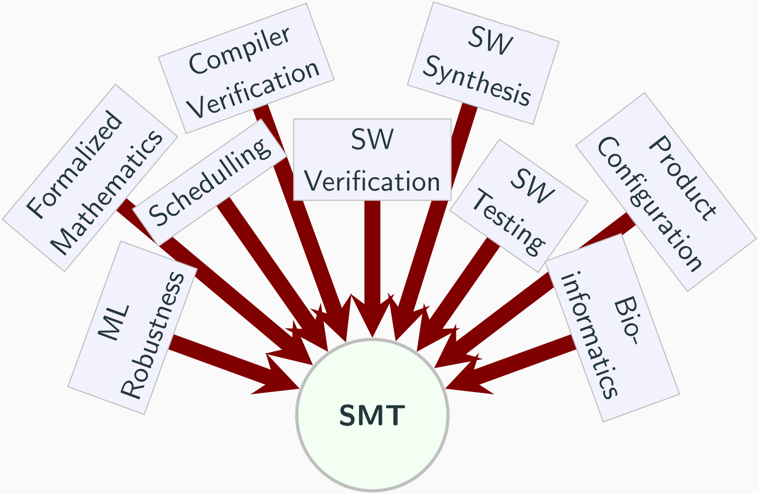
How is SMT Used in SW Verification



How is SMT Used in SW Verification



SMT Impact


Formalized Mathematics
Compiler Verification
Schedulling
SW Verification
SW Synthesis
SW Testing
Product Configuration
ML Robustness
Bio-informatics
SMT

Advertisement:

Hiring postdocs and PhD students to work on SMT + ML.

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:

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

- Alternatively, use equality:

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

Quantification and Two-player Games

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

Quantification and Two-player Games

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

  Example $\forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \vee e_1) \wedge (u_2 \vee \neg e_2)$

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 $\exists$.

Quantification and Two-player Games

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

  Example $\forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \vee e_1) \wedge (u_2 \vee \neg e_2)$
- A QBF represents a two-player game between $\forall$ and $\exists$.
- $\forall$ wins a game if the matrix becomes false.

Quantification and Two-player Games

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

  Example $\forall u_1 u_2 \exists e_1 e_2 . (\neg u_1 \vee e_1) \wedge (u_2 \vee \neg e_2)$
- 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.

Quantification and Two-player Games

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

  Example $\forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \vee e_1) \wedge (u_2 \vee \neg e_2)$
- 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 $\forall$.

Quantification and Two-player Games

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

  Example $\forall u_1 u_2 \exists e_1 e_2 . (\neg u_1 \vee e_1) \wedge (u_2 \vee \neg e_2)$
- 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 $\exists$.

Quantification and Two-player Games

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

  Example $\forall u_1 u_2 \exists e_1 e_2. (\neg u_1 \vee e_1) \wedge (u_2 \vee \neg e_2)$
- 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

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

Solving by CEGAR Expansion

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

Can be solved by $\texttt{SAT}\left(\bigwedge_{\mu \in 2^{\mathcal{U}}} \phi[\mu]\right)$. 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: [J. and Marques-Silva, 2011]

- Pick τ_0 arbitrary 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: [J. and Marques-Silva, 2011]

- Pick τ_0 arbitrary assignment to $\mathcal{E}$
- $\texttt{SAT}(\neg\phi[\tau_0]) = \mu_0$ 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: [J. and Marques-Silva, 2011]

- Pick τ_0 arbitrary assignment to $\mathcal{E}$
- $\texttt{SAT}(\neg\phi[\tau_0]) = \mu_0$ assignment to $\mathcal{U}$
- $\texttt{SAT}(\phi[\mu_0]) = \tau_1$ 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: [J. and Marques-Silva, 2011]

- Pick τ_0 arbitrary assignment to $\mathcal{E}$
- $\texttt{SAT}(\neg\phi[\tau_0]) = \mu_0$ assignment to $\mathcal{U}$
- $\texttt{SAT}(\phi[\mu_0]) = \tau_1$ assignment to $\mathcal{E}$
- $\texttt{SAT}(\neg\phi[\tau_1]) = \mu_2$ 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: [J. and Marques-Silva, 2011]

- Pick τ_0 arbitrary assignment to $\mathcal{E}$
- $\texttt{SAT}(\neg\phi[\tau_0]) = \mu_0$ assignment to $\mathcal{U}$
- $\texttt{SAT}(\phi[\mu_0]) = \tau_1$ assignment to $\mathcal{E}$
- $\texttt{SAT}(\neg\phi[\tau_1]) = \mu_2$ assignment to $\mathcal{U}$
- $\texttt{SAT}(\phi[\mu_0] \wedge \phi[\mu_1]) = \tau_2$ 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: [J. and Marques-Silva, 2011]

- Pick τ_0 arbitrary assignment to $\mathcal{E}$
- $\mathtt{SAT}(\neg\phi[\tau_0]) = \mu_0$ assignment to $\mathcal{U}$
- $\mathtt{SAT}(\phi[\mu_0]) = \tau_1$ assignment to $\mathcal{E}$
- $\mathtt{SAT}(\neg\phi[\tau_1]) = \mu_2$ assignment to $\mathcal{U}$
- $\mathtt{SAT}(\phi[\mu_0] \wedge \phi[\mu_1]) = \tau_2$ 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. [J. et al., 2012].

Function $\texttt{Solve}(\exists X \forall Y.\, \phi)$

$\alpha \leftarrow \text{true}$ // start with an empty abstraction

while true **do**

  $\tau \leftarrow \texttt{SAT}(\alpha)$ // find a candidate

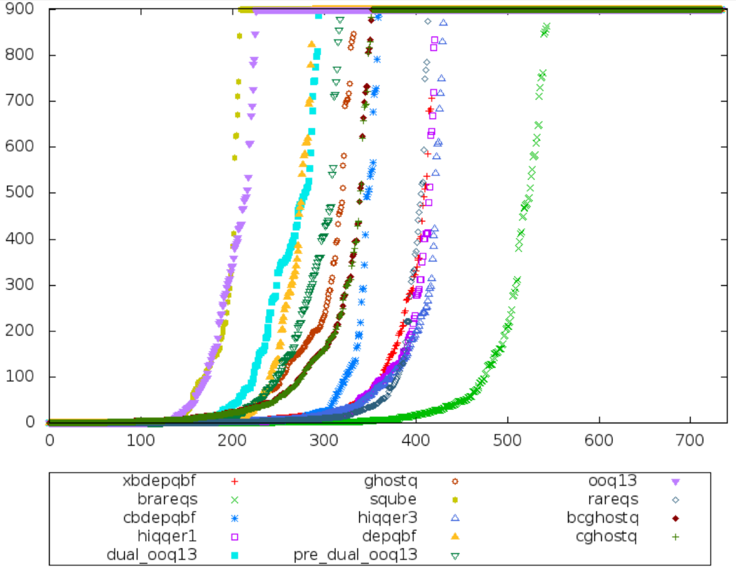
  if $\tau = \bot$ **then return** $\bot$

  $\mu \leftarrow \texttt{Solve}(\neg\phi[X \leftarrow \tau])$ // 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.

Careful Expansion: Good Example

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

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

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

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. $\mathtt{SAT}(\neg(1 \Leftrightarrow y)) \ldots y \leftarrow 0$ countermove

Careful Expansion: Bad Example

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

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

Careful Expansion: Bad Example

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

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

Careful Expansion: Bad Example

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

1. $x \leftarrow 1$ candidate
2. $\mathtt{SAT}(\neg(1 \Leftrightarrow y)) \ldots y \leftarrow 0$ countermove
3. $\mathtt{SAT}(x \Leftrightarrow 0) \ldots x \leftarrow 0$ candidate
4. $\mathtt{SAT}(\neg(0 \Leftrightarrow y)) \ldots y \leftarrow 1$ countermove
5. $\mathtt{SAT}(x \Leftrightarrow 0 \wedge x \Leftrightarrow 1) \ldots$ 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. $\texttt{SAT}(\neg(0 \Leftrightarrow y_1 \vee \neg 0 \Leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1$

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

Careful Expansion: Ugly Example

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

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

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

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

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

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

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

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

[J., 2018]

1. Enumerate some number of candidate–countermove pairs.

Use Machine Learning

[J., 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

[J., 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.

Use Machine Learning

[J., 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.

Machine Learning Example

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

Machine Learning Example

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 $i \in 2..n$.

Machine Learning Example

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 $i \in 2..n$.
- $SAT(x_1 \Leftrightarrow \neg x_1 \vee \bigvee_{i \in 2..n} x_2 \Leftrightarrow 1)$

Machine Learning Example

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: $y_1 \leftarrow \neg x_1$ $y_2 \leftarrow \neg x_2$ ...

Machine Learning Example

x_1	x_2	$\dots$	x_n	y_1	y_2	$\dots$	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: $y_1 \leftarrow \neg x_1$ $y_2 \leftarrow \neg x_2$ $\dots$
- 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.

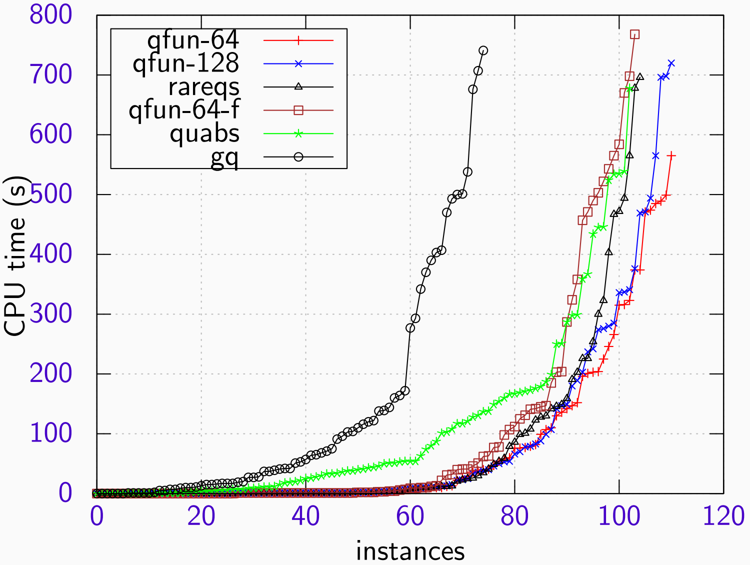
Current Implementation

- 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

- 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.
- Additional heuristic: If a learned function still works, keep it. "Don't fix what ain't broke."

Current Implementation: Experiments



Towards SMT

Towards SMT

- in SMT we always have equality
- almost always need uninterpreted functions
- Challenge: learning uninterpreted functions
- looking at finite models

$$(\forall x)(\mathsf{range}(x) \rightarrow \mathsf{memory}(x) = c)$$

Bernays-Schönfinkel (EPR)

$$\forall X.\ \phi$$

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

Bernays-Schönfinkel (EPR)

$$\forall X.\ \phi$$

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

Bernays-Schönfinkel (EPR)

$$\forall X.\ \phi$$

- 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 $\leq n$.

Bernays-Schönfinkel (EPR)

$$\forall X.\ \phi$$

- 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 $*_1, \ldots, *_{n'}$, $n' \leq n$.

CEGAR for Finite Models

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

p_i predicates, c_i constants, X variables

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

CEGAR for Finite Models

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

p_i predicates, c_i constants, X variables

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

CEGAR for Finite Models

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

p_i predicates, c_i constants, X variables

1. $\alpha \leftarrow \texttt{true}$
2. Find interpretation for α: $\mathcal{I} \leftarrow \texttt{SAT}(\alpha)$
3. If no interpretation, formula is *false*. STOP.

CEGAR for Finite Models

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

p_i predicates, c_i constants, X variables

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

CEGAR for Finite Models

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

p_i predicates, c_i constants, X variables

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

CEGAR for Finite Models

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

p_i predicates, c_i constants, X variables

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

CEGAR for Finite Models

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

p_i predicates, c_i constants, X variables

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

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,

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.

Learning in Finite Models' CEGAR

1. Consider some finite grounding:
   $\exists p_1 \dots p_m \exists c_1 \dots c_n \bigwedge_{\mu \in \omega} . \ \phi[\mu]$
   p_i predicates, c_i constants,
2. Calculate interpretation by e.g. Ackermanization.
3. The interpretation only matters on the existing ground terms.

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.
4. *Learn* entire interpretation from observing values of existing terms.

Learning in Finite Models' CEGAR, Example

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

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 $\{X_1 \triangleq *_1, X_1 \triangleq *_0 \ldots X_n \triangleq *_0\}$:

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

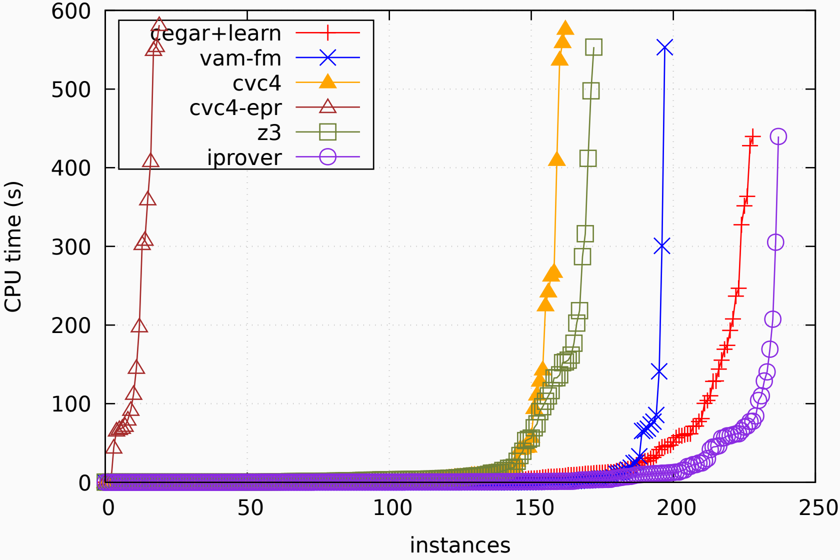
Learning in Finite Models' CEGAR, Example

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:
   $t \triangleq *_1$
   $p(*_0 \ldots, *_0) \triangleq \text{False}$
   $p(*_1 \ldots, *_0) \triangleq \text{True}$

Learning in Finite Models' CEGAR, Example

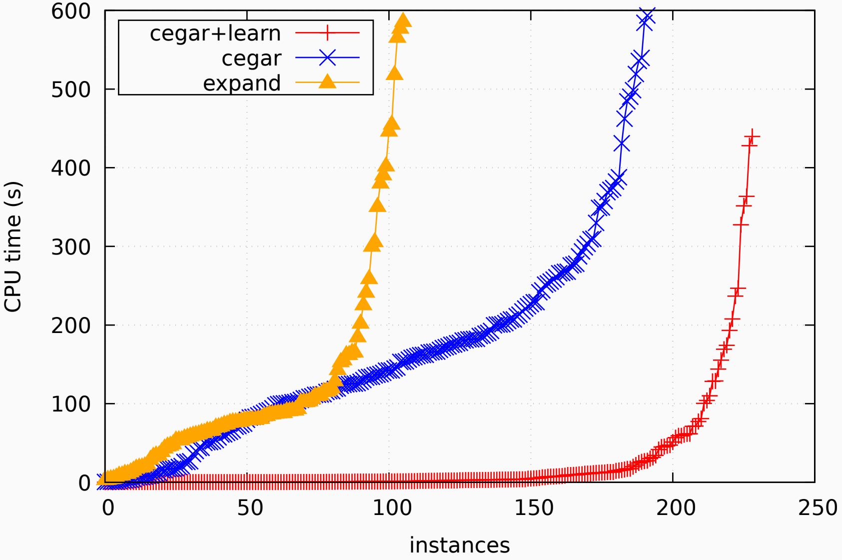
1. $\forall X.\ p(X_1, \ldots, X_n) \Leftrightarrow (X_1 = t)$
2. Ground by $\{X_i \triangleq *_0\}$ and $\{X_1 \triangleq *_1, X_1 \triangleq *_0 \ldots X_n \triangleq *_0\}$:
3. $(p(*_0, \ldots, *_0) \Leftrightarrow *_0 = t) \wedge (p(*_1, \ldots, *_0) \Leftrightarrow *_1 = t)$
4. Partial interpretation: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

Results EPR

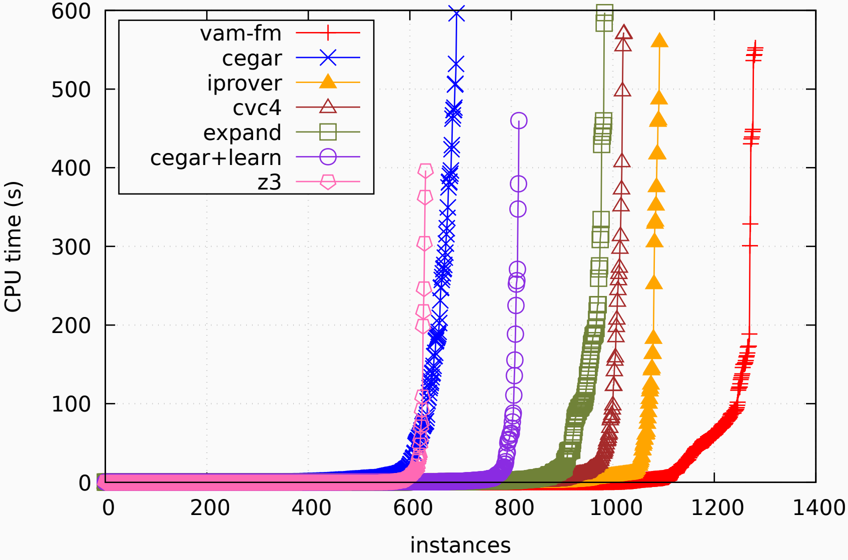


600
500
400
300
200
100
0
CPU time (s)
0
50
100
150
200
250
instances
cegar+learn
vam-fm
cvc4
cvc4-epr
z3
iprover

Results EPR: QFM



Results SAT NON-EPR



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:

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

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 $\ldots\exists\mathbb{B}^n\forall\mathbb{B}^m\ldots$, 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 $\ldots \exists \mathbb{B}^n \forall \mathbb{B}^m \ldots$, 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 $D_1 \times \cdots \times D_k \to \mathbb{B}$
- 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:

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

- 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?

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?

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

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

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