JOHANNES KEPLER
UNIVERSITY LINZ

PyQBF
A Python Framework for Solving Quantified Boolean Formulas

Mark Peyrer, Maximilian Heisinger, Martina Seidl

**JOHANNES KEPLER
UNIVERSITY LINZ**
Altenberger Straße 69
4040 Linz, Austria
jku.at

Introduction

This is a QBF

Example

$$\underbrace{\forall x_1 \exists x_2 \exists x_3 \forall x_4 \cdots \exists x_n :}_{\text{Prefix}}$$

$$\underbrace{\begin{array}{r} (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \wedge \\ (\neg x_1 \vee x_4 \vee \neg x_9 \vee x_{11}) \wedge \\ (\neg x_7 \vee \neg x_5 \vee \neg x_{10}) \wedge \\ \dots \end{array}}_{\textit{Propositional Formula}}$$

This is a QBF

Example

$$\underbrace{\forall x_1 \exists x_2 \exists x_3 \forall x_4 \cdots \exists x_n :}_{\textbf{Prefix}}$$

$$\underbrace{\begin{array}{r} (x_1 \lor x_2 \lor x_3 \lor x_4 \lor x_5) \land \\ (\neg x_1 \lor x_4 \lor \neg x_9 \lor x_{11}) \land \\ (\neg x_7 \lor \neg x_5 \lor \neg x_{10}) \land \\ \ldots \end{array}}_{\textbf{Propositional Formula}}$$

Two Parts

This is a QBF

Example

$$\underbrace{\forall x_1 \exists x_2 \exists x_3 \forall x_4 \cdots \exists x_n :}_{\text{Prefix}}$$

Boolean Variables

$$\underbrace{\begin{array}{r}
(x_1 \lor x_2 \lor x_3 \lor x_4 \lor x_5) \land \\
(\neg x_1 \lor x_4 \lor \neg x_9 \lor x_{11}) \land \\
(\neg x_7 \lor \neg x_5 \lor \neg x_{10}) \land \\
\cdots
\end{array}}_{\textit{Propositional Formula}}$$

This is a QBF

Example

$$\underbrace{\forall x_1 \exists x_2 \exists x_3 \forall x_4 \cdots \exists x_n :}_{\text{Prefix}}$$

Quantifiers

$$\underbrace{\begin{array}{r}(x_1 \lor x_2 \lor x_3 \lor x_4 \lor x_5)\land \\ (\neg x_1 \lor x_4 \lor \neg x_9 \lor x_{11})\land \\ (\neg x_7 \lor \neg x_5 \lor \neg x_{10})\land \\ \dots\end{array}}_{\textit{Propositional Formula}}$$

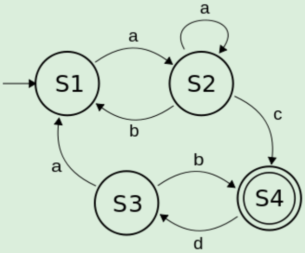
This is a QBF

Example

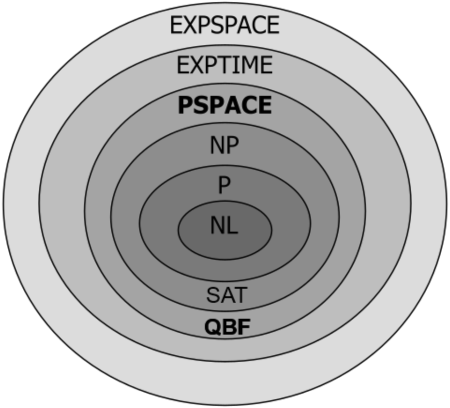
$$\underbrace{\forall x_1 \exists x_2 \exists x_3 \forall x_4 \cdots \exists x_n :}_{\text{Prefix}}$$

$$\underbrace{\begin{array}{r} (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5)\wedge \\ (\neg x_1 \vee x_4 \vee \neg x_9 \vee x_{11})\wedge \\ (\neg x_7 \vee \neg x_5 \vee \neg x_{10})\wedge \\ \dots \end{array}}_{\textit{Propositional Formula}}$$

e.g. used in Bounded Model Checking



QBF complexity



”Propositional logic + ∀ and ∃”

It’s sunny in the QBF-World...

- Quantified Boolean Formulas (QBFs) are an important research topic

It’s sunny in the QBF-World…

- Quantified Boolean Formulas (QBFs) are an important research topic
- Regularly we have the QBF-gallery with new formulas and solvers

It’s sunny in the QBF-World…

- Quantified Boolean Formulas (QBFs) are an important research topic
- Regularly we have the QBF-gallery with new formulas and solvers
- A lot of improvement can be seen

...with a little bit of rain

- Only few QBF-Solver have an API

...with a little bit of rain

- Only few QBF-Solver have an API
- Domain specific knowledge is necessary to use them

...with a little bit of rain

- Only few QBF-Solver have an API
- Domain specific knowledge is necessary to use them
- Thus writing a prototype is hard

What we want

- Develop tools and algorithms on a higher API level

What we want

- Develop tools and algorithms on a higher API level
- Make it easy to start working with QBFs

What we want

- Develop tools and algorithms on a higher API level
- Make it easy to start working with QBFs
- Allow rapid prototyping

What we want

- Develop tools and algorithms on a higher API level
- Make it easy to start working with QBFs
- Allow rapid prototyping

PyQBF

PySAT

Not a new Concept

- In 2018, Ignatiev et al. released a paper[1] identifying the problems with SAT-solvers

[1] Ignatiev, A., Morgado, A., and Marques-Silva, J. (2018, June). PySAT: A Python toolkit for prototyping with SAT oracles. In International Conference on Theory and Applications of Satisfiability Testing (pp. 428-437). Cham: Springer International Publishing.

Not a new Concept

- In 2018, Ignatiev et al. released a paper[1] identifying the problems with SAT-solvers
- Low-level representation required is not sufficient for state-of-the-art problems

[1] Ignatiev, A., Morgado, A., and Marques-Silva, J. (2018, June). PySAT: A Python toolkit for prototyping with SAT oracles. In International Conference on Theory and Applications of Satisfiability Testing (pp. 428-437). Cham: Springer International Publishing.

Not a new Concept

- In 2018, Ignatiev et al. released a paper[1] identifying the problems with SAT-solvers
- Low-level representation required is not sufficient for state-of-the-art problems
- They developed *PySAT*[2] — A toolkit for prototyping with SAT Oracles

[1] Ignatiev, A., Morgado, A., and Marques-Silva, J. (2018, June). PySAT: A Python toolkit for prototyping with SAT oracles. In International Conference on Theory and Applications of Satisfiability Testing (pp. 428-437). Cham: Springer International Publishing.

[2] https://pysathq.github.io/

Easy prototyping with SAT — Example

```python
from pysat.formula import CNF
from pysat.solvers import Solver

formula = CNF(from_file="./my/cnf.dimacs")
formula2 = CNF(from_aiger="./your/aiger.aag")

with Solver(name='cadical153', bootstrap_with=formula) as solver:
    solver.add_clause([1,2,3])
    solver.append_formula(formula2)
    print(solver.solve())
```

Many features

- Solving formulas
- Model enumeration
- Generation of cardinality constraints
- Preprocessing
- Certification
- ...

PyQBF

PyQBF as an extension of PySAT

PyQBF as an extension of PySAT

- Formulas extending `pysat.formula.CNF` by inheritance

PyQBF as an extension of PySAT

- Formulas extending `pysat.formula.CNF` by inheritance
- Similar interface with a lot of interoperability

PyQBF as an extension of PySAT

- Formulas extending `pysat.formula.CNF` by inheritance
- Similar interface with a lot of interoperability
- Integrating commonly used QBF-solvers and Preprocessors

PyQBF as an extension of PySAT

- Formulas extending `pysat.formula.CNF` by inheritance
- Similar interface with a lot of interoperability
- Integrating commonly used QBF-solvers and Preprocessors

Expanding the power of PySAT to the domain of QBFs!

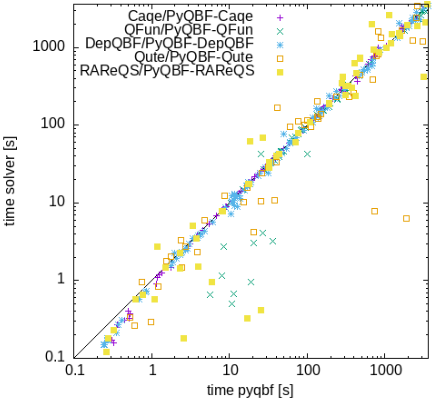
Easy prototyping with QBF — Example

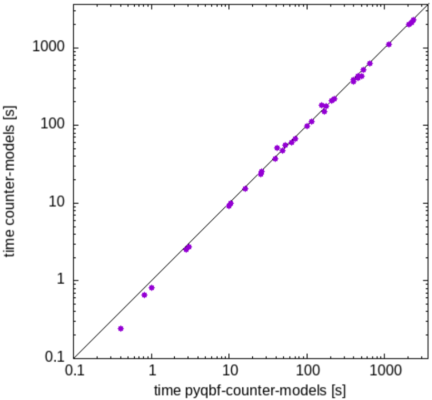
```python
from pyqbf.formula import PCNF
from pyqbf.solver import Solver

formula = PCNF(from_file="./my/cnf.qdimacs")
formula2 = PCNF(from_aiger="./your/aiger.aag")
formula2.forall(1,2,3).exists(5,6,7,8)

with Solver(name='depqbf', bootstrap_with=formula) as solver:
    solver.add_clause([1,2,3])
    solver.append_formula(formula2)
    print(solver.solve())
```

Features

- Solving formulas with 5 state-of-the-art QBF-solvers
- (Counter-)Model enumeration
- Preprocessing with Bloqqer
- Import CNFs
- Normalize formulas
- Incremental solving with non-assuming solvers using QuAPI
- ...

...and all of that with only little overhead





Availability

- *PyQBF: A Python Framework for Solving Quantified Boolean Formulas* was accepted for the 19th International Conference on Integrated Formal Methods (iFM24)
- We furthermore submitted an artifact publicly available at 10.5281/zenodo.11118824
- The current release version of PyQBF can be found at https://gitlab.sai.jku.at/qbf/pyqbf
- The documentation of the framework is available at https://qbf.pages.sai.jku.at/pyqbf/

Thank you for your attention!

Any Questions?

JKU

JOHANNES KEPLER
UNIVERSITY LINZ