AQME'10 System Description

Luca Pulina and Armando Tacchella

University of Genoa – DIST - Viale Causa 13 – 16145 Genoa (Italy)

POS 2010 - Edinburgh, July 10, 2010

What is a quantified Boolean formula?

Consider a Boolean formula, e.g.,

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$$

Adding existential "$\exists$" and universal "$\forall$" quantifiers, e.g.,

$$\forall x_1 \exists x_2 (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$$

yields a quantified Boolean formula (QBF).

What is the meaning of a QBF?

A QBF, e.g.,

$$\forall x_1 \exists x_2 (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$$

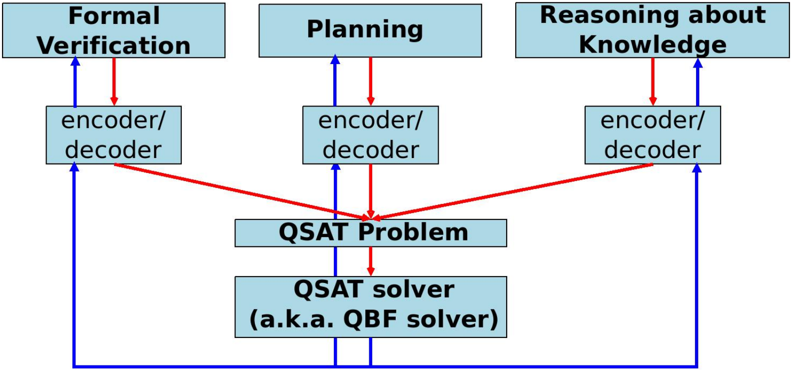
is true if and only if

for every value of x_1 there exist a value of x_2 such that $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ is propositionally satisfiable

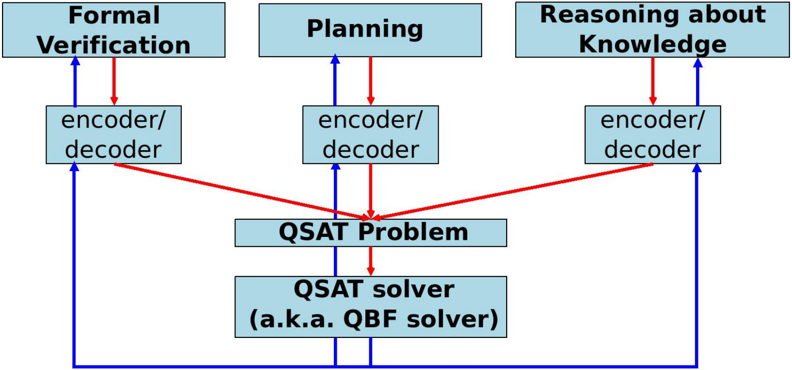
Given any QBF ψ:

- if $\psi = \forall x \varphi$ then ψ is true iff $\varphi_{|_{x=0}} \wedge \varphi_{|_{x=1}}$ is true
- if $\psi = \exists x \varphi$ then ψ is true iff $\varphi_{|_{x=0}} \vee \varphi_{|_{x=1}}$ is true

QBFs as a logic "assembly" language

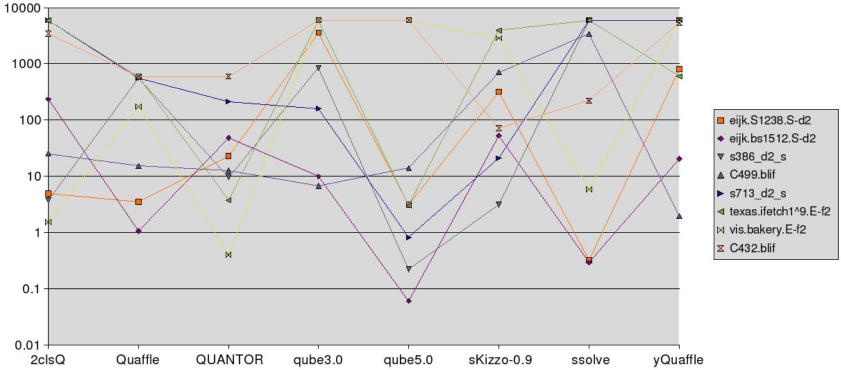


QBFs as a logic “assembly” language

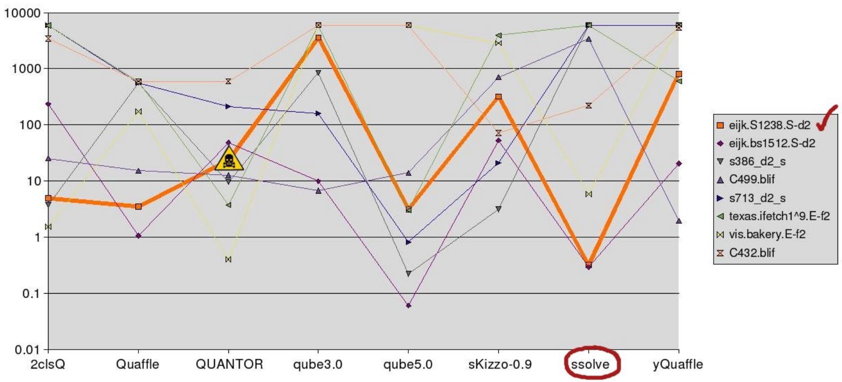


This approach works fine as long as QBF solvers are robust!

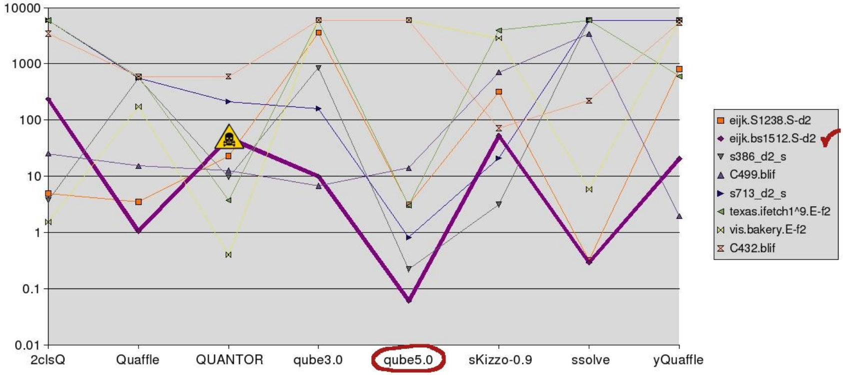
Are state-of-the-art QBF solvers robust?



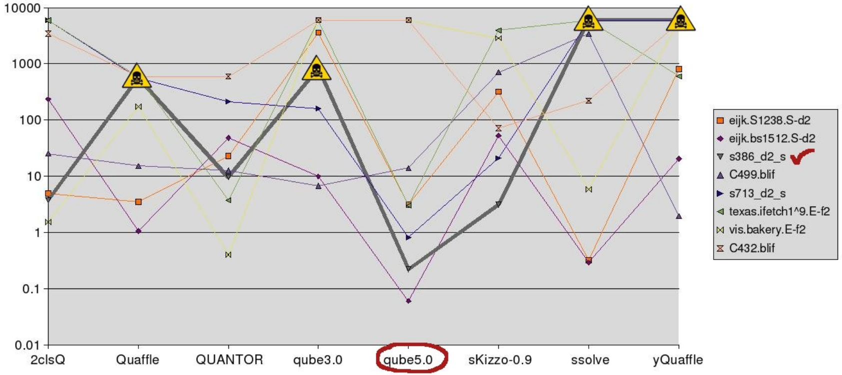
Are state-of-the-art QBF solvers robust?



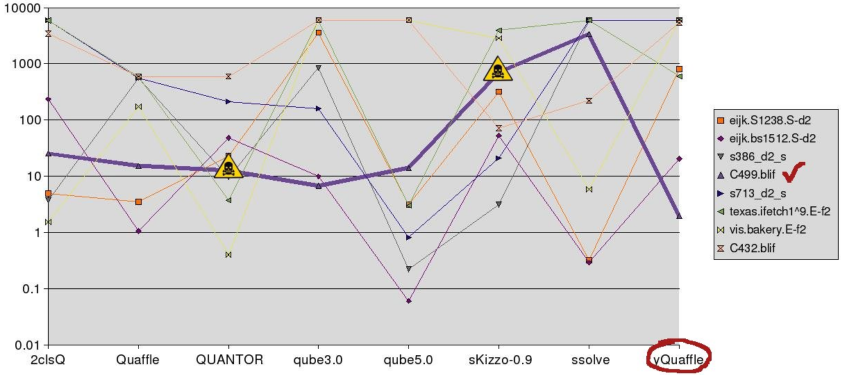
Are state-of-the-art QBF solvers robust?



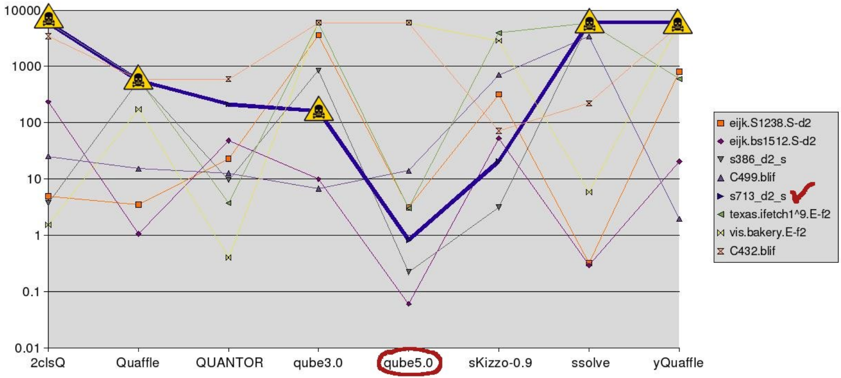
Are state-of-the-art QBF solvers robust?



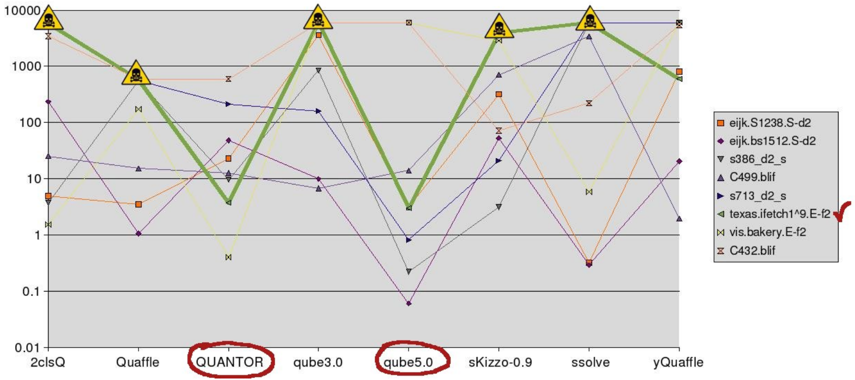
Are state-of-the-art QBF solvers robust?



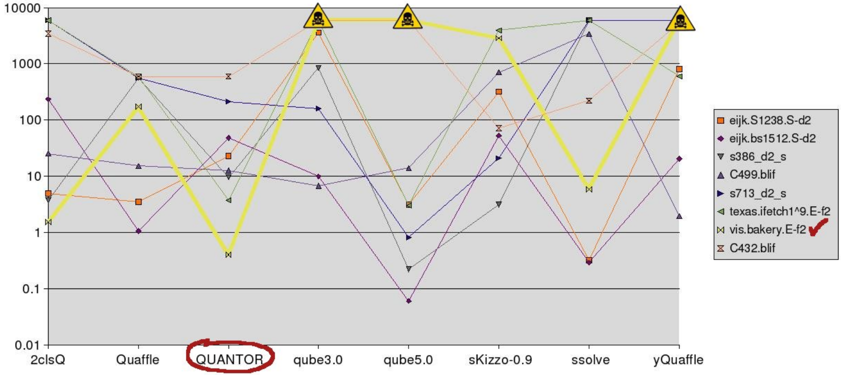
Are state-of-the-art QBF solvers robust?



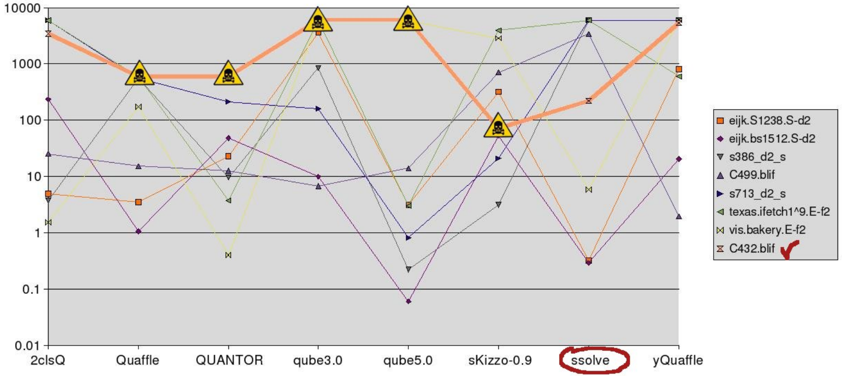
Are state-of-the-art QBF solvers robust?



Are state-of-the-art QBF solvers robust?



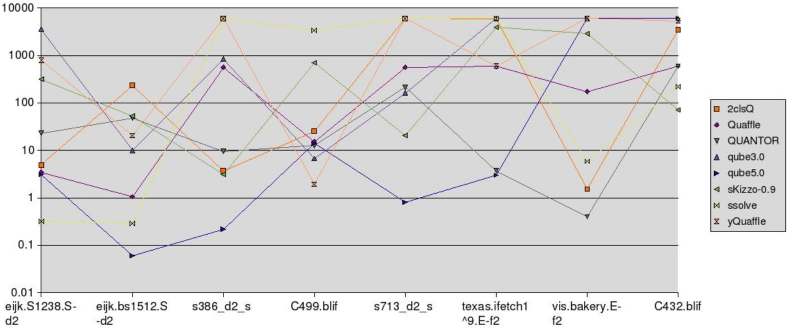
Are state-of-the-art QBF solvers robust?



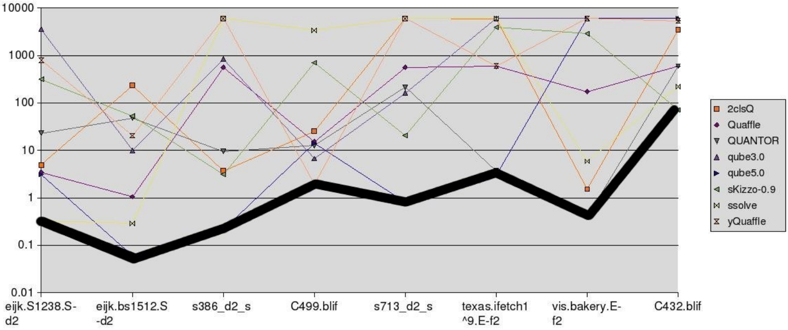
Are state-of-the-art QBF solvers robust?

NOT REALLY...

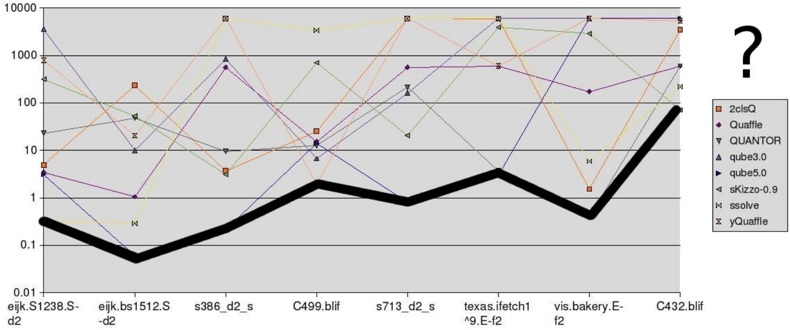
Goal: a robust QBF solver



Goal: a robust QBF solver



Goal: a robust QBF solver



Outline

1. Engineering a robust QBF solver
2. Designing a self-adaptive multi-engine
3. Experiments
4. Conclusions & future work

Outline

Two approaches to yield a robust solver

Brute force

Given *m* QSAT instances and *n* solvers (engines)

1. Run each engine on a separate machine.
2. Stop all the engines as soon as one solves the instance, or all the engines exhaust resources.
3. Continue with the next instance (if any).

Two approaches to yield a robust solver

Brute force

Given *m* QSAT instances and *n* solvers (engines)

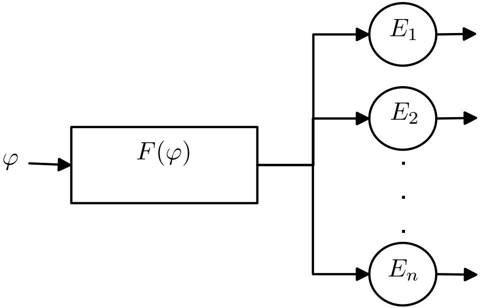
1. Run each engine on a separate machine.
2. Stop all the engines as soon as one solves the instance, or all the engines exhaust resources.
3. Continue with the next instance (if any).

Intelligence

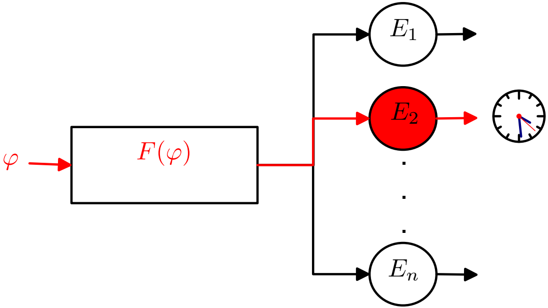
Understand which engine is best for which QBFs

- Fairly old idea: asset allocation in economics.
- Looking for dynamically adaptive policies.
- Algorithm portfolios: SAT, SMT, QBFs (see related work).

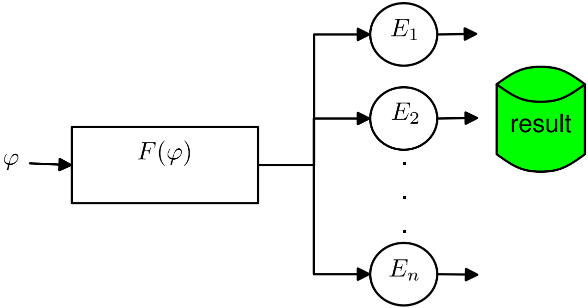
Intelligence = Learning (to choose engines)



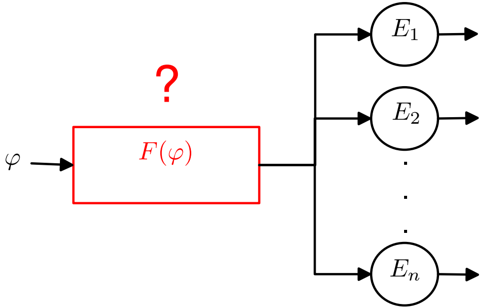
Intelligence = Learning (to choose engines)



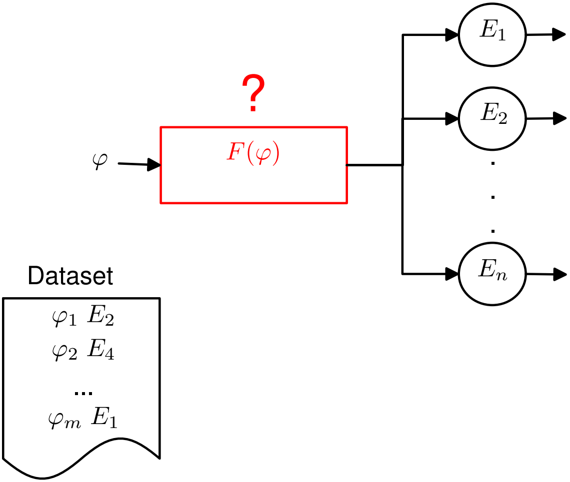
Intelligence = Learning (to choose engines)



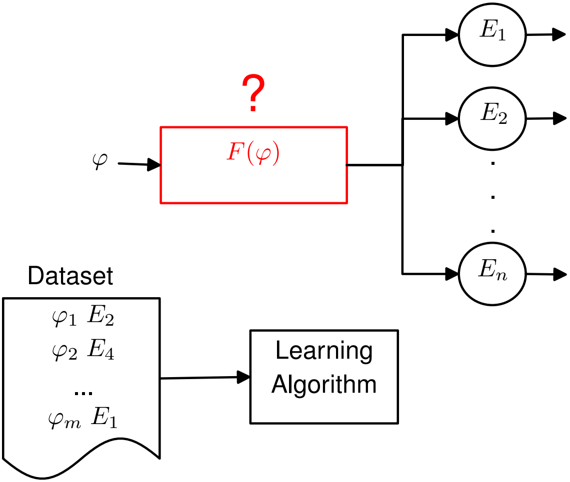
Intelligence = Learning (to choose engines)



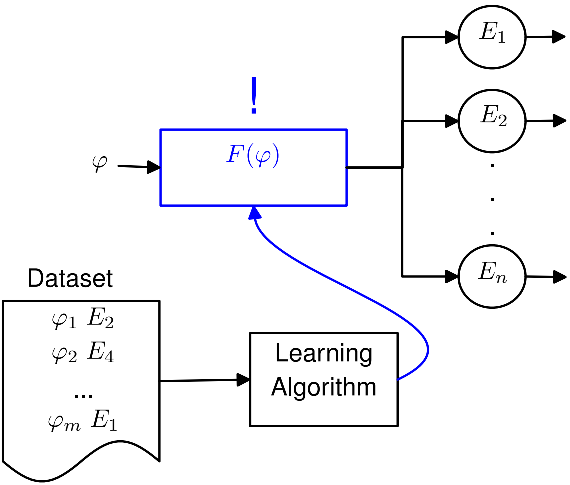
Intelligence = Learning (to choose engines)



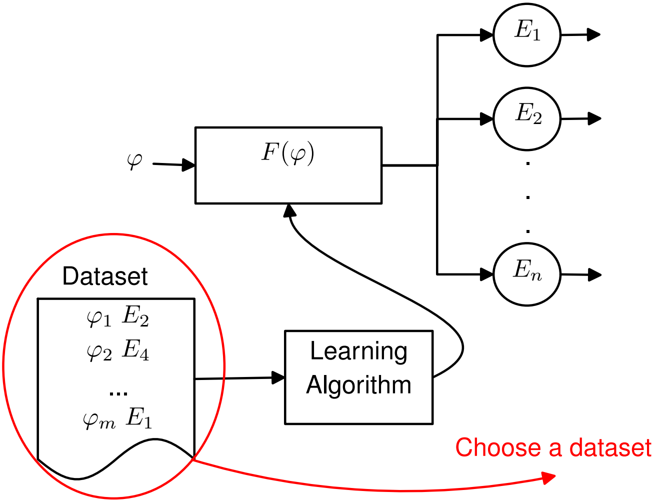
Intelligence = Learning (to choose engines)



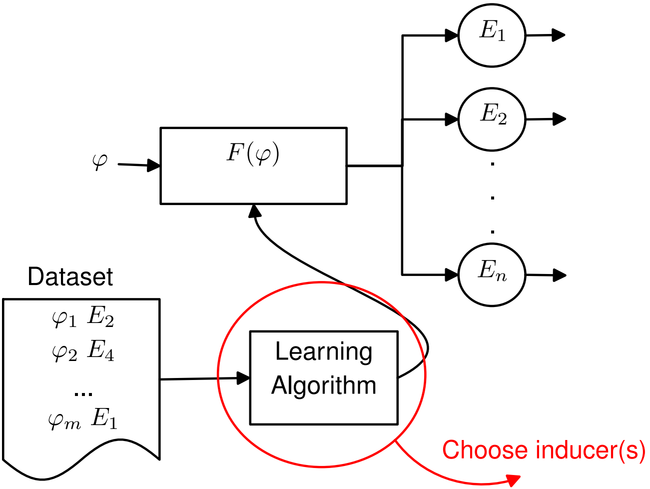
Intelligence = Learning (to choose engines)



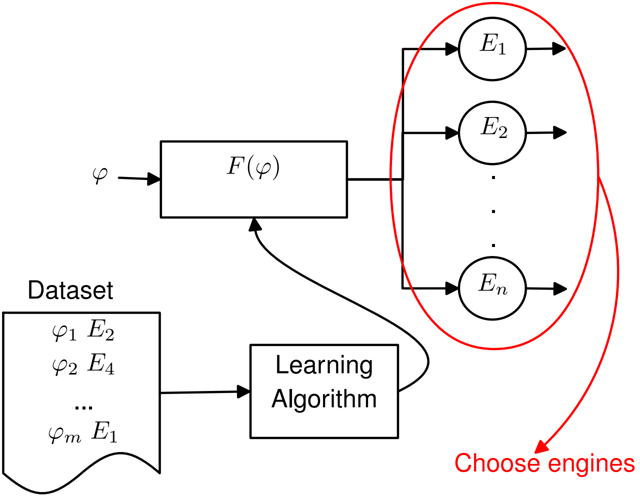
Intelligence = Learning (to choose engines)



Intelligence = Learning (to choose engines)



Intelligence = Learning (to choose engines)



Choosing datasets

- QBFLIB (`www.qbflib.org`), a repository of QBFs
  - More than 15K formulas in a standard format.
  - Artificially generated, toy problems, realistic encodings, challenge problems, ...
- QBF solvers competitions (`www.qbfeval.org`)
  - A subset of the formulas available in QBFLIB.
  - Up-to-date performance data about QBF solvers.

Choosing datasets

- QBFLIB (`www.qbflib.org`), a repository of QBFs
  - More than 15K formulas in a standard format.
  - Artificially generated, toy problems, realistic encodings, challenge problems, ...
- QBF solvers competitions (`www.qbfeval.org`)
  - A subset of the formulas available in QBFLIB.
  - Up-to-date performance data about QBF solvers.

Our choice in AQME'10

The whole QBFEVAL'08 dataset (3326 fixed structured formulas).

Representing QBFs

Basic features regarding:

- Clauses: total number, number of Horn clauses, . . .
- Variables: total number, existential and universal, . . .
- Quantifiers: alternations, . . .
- Literals: total number, average per clause, . . .
- . . .

Combined features: ratios/products between basic features.

Representing QBFs

Basic features regarding:

- Clauses: total number, number of Horn clauses, . . .
- Variables: total number, existential and universal, . . .
- Quantifiers: alternations, . . .
- Literals: total number, average per clause, . . .
- . . .

Combined features: ratios/products between basic features.

Our choice in AQME'10

109 cheap syntactic features for each QBF.

Choice of inductive models

Our desiderata:

- Deal with numerical attributes (QBF features) and multiple class labels (engines).
- No assumptions of normality or (in)dependence among the features.
- No complex parameter tuning, thanks!

Choice of inductive models

Our desiderata:

- Deal with numerical attributes (QBF features) and multiple class labels (engines).
- No assumptions of normality or (in)dependence among the features.
- No complex parameter tuning, thanks!

Our choice in AQME'10

Nearest-neighbour (1-NN)

- We also implemented multivariate logistic regression, decision trees, and decision rules.
- We select 1-NN for its robustness w.r.t. the inductive models above (see [Pulina and Tacchella, CP-DP'08]).

Choosing reasoning engines

- QBFEVALs reveal major differences between
  - Heuristic search based solvers.
  - Hybrid solvers mainly based on other techniques (e.g., resolution, skolemization), but possibly including search.
- Which solvers to choose as basic engines?
  - Only the best “search” and “hybrid”?
  - All state of the art solvers?
  - Something in between?

Choosing reasoning engines

- QBFEVALs reveal major differences between
  - Heuristic search based solvers.
  - Hybrid solvers mainly based on other techniques (e.g., resolution, skolemization), but possibly including search.
- Which solvers to choose as basic engines?
  - Only the best "search" and "hybrid"?
  - All state of the art solvers?
  - Something in between?

Our selection in AQME'10

- Search-based: QUBE3.1, SSOLVE-UT, and 2CLSQ.
- Hybrid: QUANTOR2.11, and SKIZZO-0.9-STD.

Choosing reasoning engines

- QBFEVALs reveal major differences between
  - Heuristic search based solvers.
  - Hybrid solvers mainly based on other techniques (e.g., resolution, skolemization), but possibly including search.
- Which solvers to choose as basic engines?
  - Only the best "search" and "hybrid"?
  - All state of the art solvers?
  - Something in between?

Our selection in AQME'10

- Search-based: QuBE3.1, SSOLVE-UT, and 2CLSQ.
- Hybrid: QUANTOR2.11, and sKIZZO-0.9-STD.

"Vintage engines" offer us a baseline to compare the current progress in the development of QBF solvers.

Outline

Designing a self-adaptive multi-engine

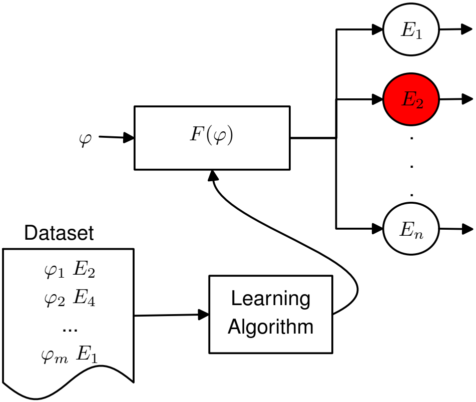
How could AQME'10 learn by its incorrect predictions?

Designing a self-adaptive multi-engine

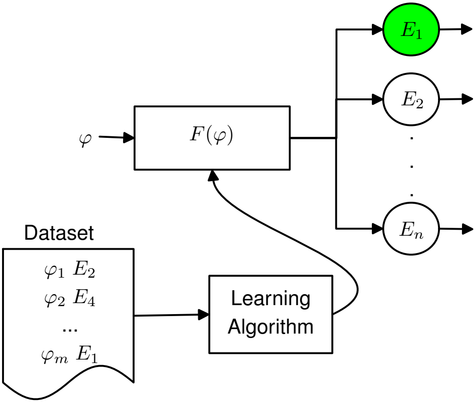
How could AQME'10 learn by its incorrect predictions?

Retraining: adaptation schema applied to engine selection policies whenever they fail to give good predictions.

Retraining

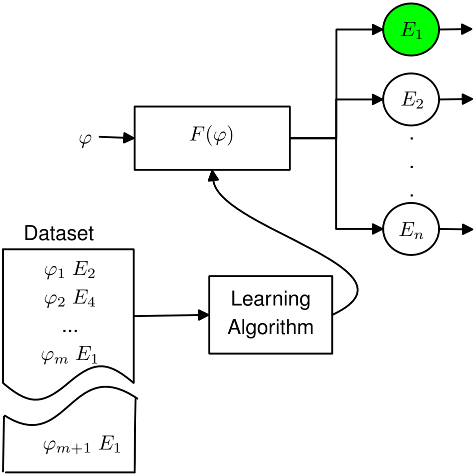

φ
$F(\varphi)$
E_1
E_2
E_n
Dataset
φ_1 E_2
φ_2 E_4
...
φ_m E_1
Learning Algorithm

Retraining

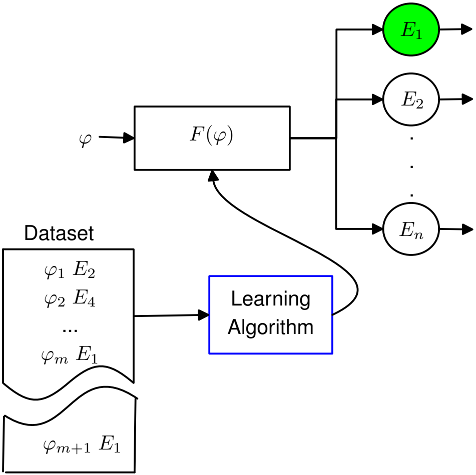


φ
$F(\varphi)$
E_1
E_2
E_n
Dataset
φ_1 E_2
φ_2 E_4
...
φ_m E_1
Learning
Algorithm

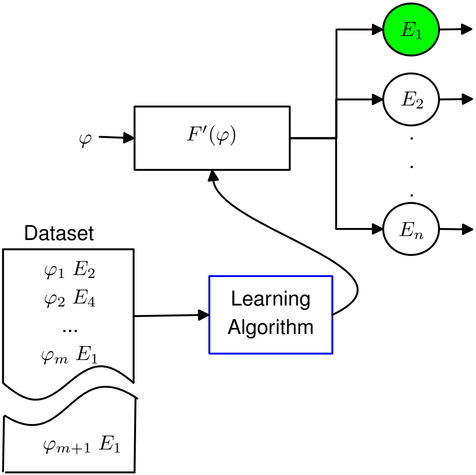
Retraining


E_1
E_2
φ
$F(\varphi)$
E_n
Dataset
φ_1 E_2
φ_2 E_4
...
φ_m E_1
Learning Algorithm
φ_{m+1} E_1

Retraining


φ
$F(\varphi)$
E_1
E_2
E_n
Dataset
φ_1 E_2
φ_2 E_4
...
φ_m E_1
φ_{m+1} E_1
Learning Algorithm

Retraining



φ
$F'(\varphi)$
E_1
E_2
E_n
Dataset
φ_1 E_2
φ_2 E_4
...
φ_m E_1
φ_{m+1} E_1
Learning Algorithm

Retraining policies

Critical points for AQME'10 performances:

- How much CPU time is granted to each engine.
- Which engine is called for retraining.

Retraining policies

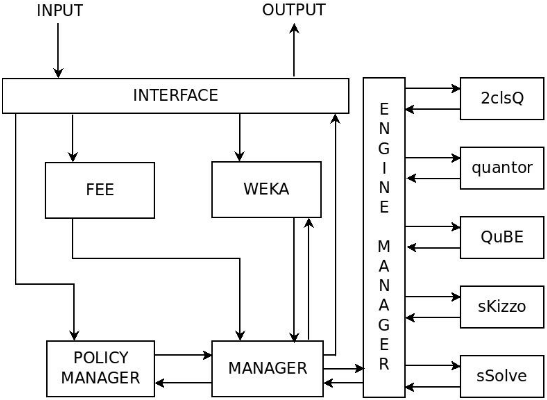
Critical points for AQME'10 performances:

- How much CPU time is granted to each engine.
- Which engine is called for retraining.

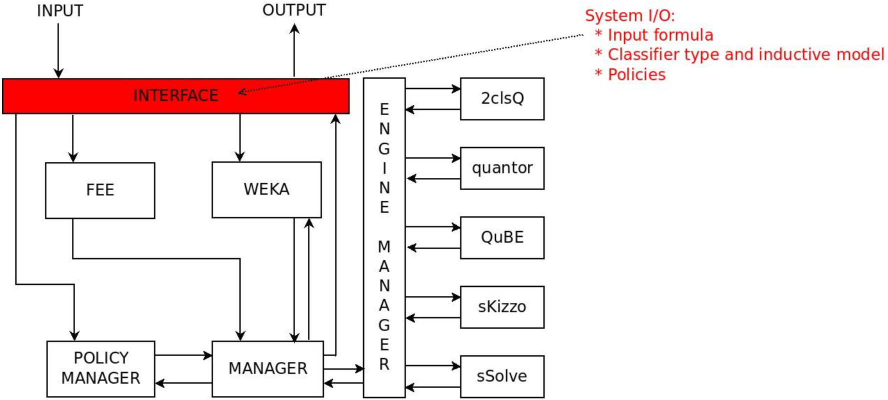
Policies in AQME'10

- **Granted CPU time**: "Trust the Predicted Engine"
  - A fixed amount of CPU time is granted to the predicted solver.
  - If it fails, another engine is called (following the engine selection policy), with a granted amount of CPU time until the solver solves the input formula.
  - If the formula is not solved, the originally predicted engine is fired, with the time limit assigned to the remaining time.
- **Engine selection**: The engine to fire is selected according to the QBFEVAL'06 ranking.

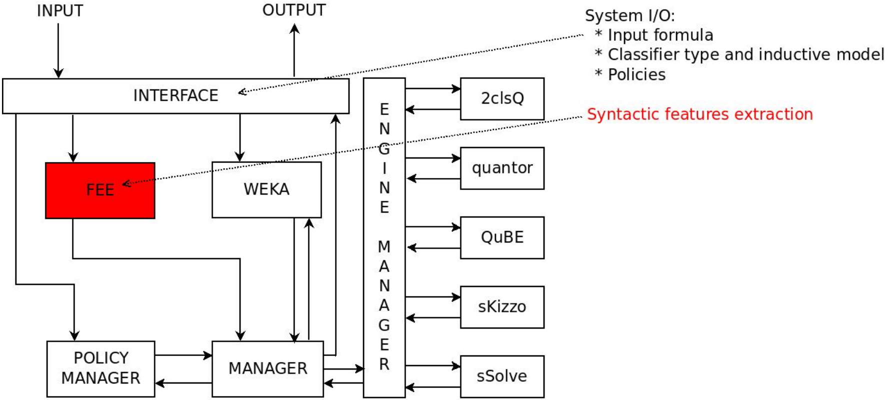
AQME'10 architecture



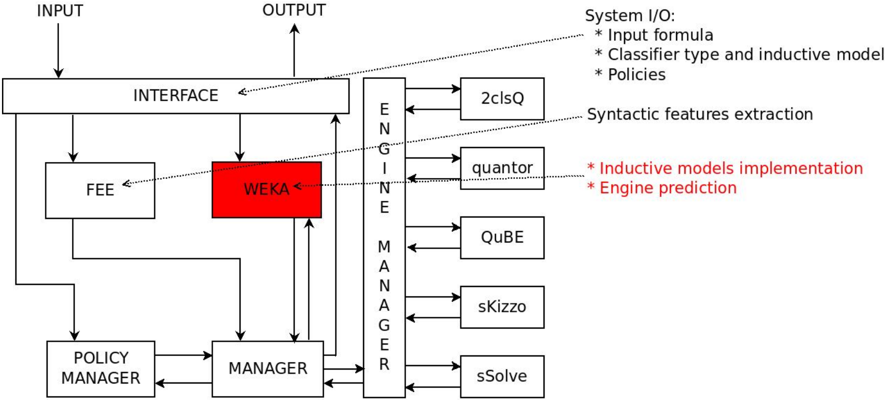
AQME'10 architecture



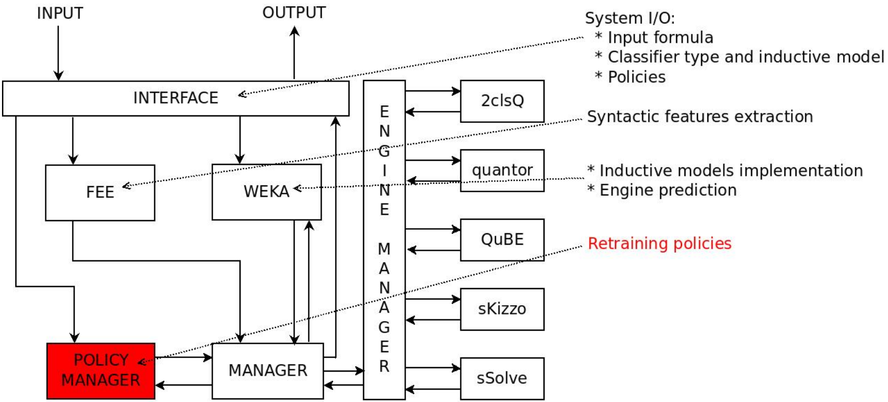
AQME'10 architecture



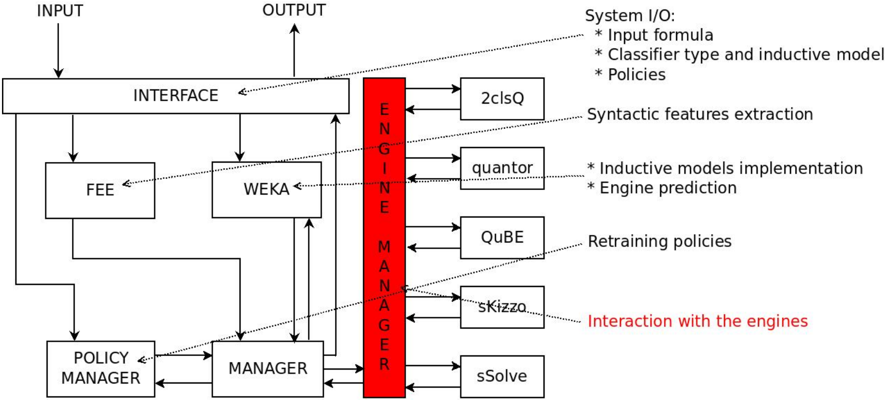
AQME'10 architecture



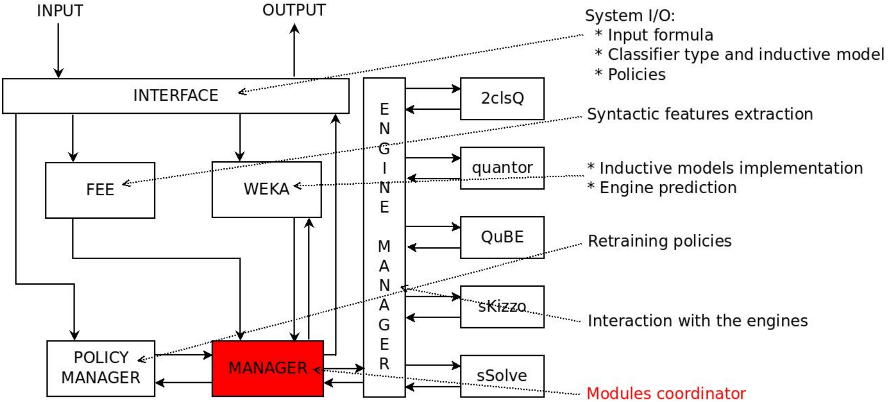
AQME'10 architecture



AQME'10 architecture



AQME'10 architecture



Outline

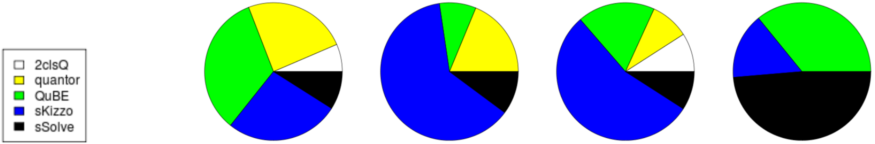
AQME'10@QBFEVAL'10

Solver	MAIN		2QBF		SH		RND	
	#	Time	#	Time	#	Time	#	Time
AIGSOLVE	329	22786.60	NA	NA	**37**	1140.01	NA	NA
AQME'10	**434**	33346.60	128	2323.11	11	30132.40	**407**	20078.90
DEPQBF	370	21515.30	24	690.42	4	41448.00	342	12895.10
DEPQBF-PRE	356	18995.90	51	877.02	4	33371.90	343	9438.62
NENOFEX	225	13786.90	50	3545.65	3	30194.20	149	34502.80
QMAIGA	361	43058.10	NA	NA	NA	NA	NA	NA
QUANTOR3.1	205	6711.37	48	3689.30	5	57960.90	134	2830.97
STRUQS'10	240	32839.70	**132**	1399.30	5	26257.30	117	15480.40

- Best[1] solver in MAIN and RND tracks.
- Good performance in 2QBF and SH tracks.

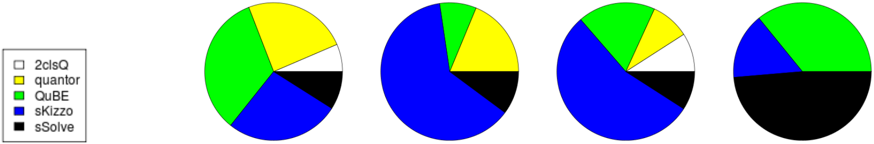
[1] In the sense of numbers of problems solved within the CPU time limit

Looking inside AQME'10



	MAIN	2QBF	SH	RND
2CLSQ	28	–	1	–
QUANTOR2.11	106	24	1	–
QUBE3.1	**145**	11	2	146
SKIZZO	116	**80**	**6**	63
SSOLVE-UT	39	13	1	**198**
Retrainings	22	3	–	15

Looking inside AQME'10



	MAIN	2QBF	SH	RND
2CLSQ	28	–	1	–
QUANTOR2.11	106	24	1	–
QUBE3.1	**145**	11	2	146
SKIZZO	116	**80**	**6**	63
SSOLVE-UT	39	13	1	**198**
Retrainings	22	3	–	15

Self-adaptation based on the characteristics of the test set.

Outline

Conclusions

- A multiengine solver is a robust alternative to current state-of-the-art QBF solvers.
- Good performance achieved also using engines date back 2006.
- Retraining algorithm increases the performances in terms of number of solved formula.
- Performances “limited” by the **State-of-the-art solver**, i.e., the ideal solver that always fares the best time among all the considered solvers.

Future work

- Mechanism for the automatic integration of new engines.
- Implementation of new learning algorithms (see, e.g., D. Stern et al., AAAI 2010).
- Integration between different algorithms, not black-box engines (see, e.g., Pulina and Tacchella, FROCOS 2009).

Thank you!