SAT-Based Methods for Circuit Synthesis*

October 22, 2014

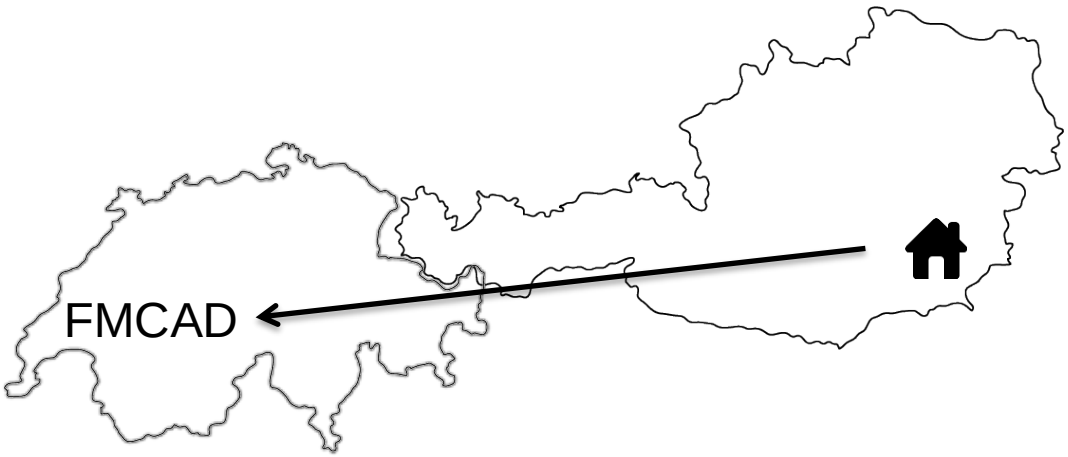
Roderick Bloem
Patrick Klampfl
Robert Könighofer

Uwe Egly
Florian Lonsing

* This work was supported in part by the Austrian Science Fund (FWF) through the national research network RiSE (S11406-N23, S11409-N23) and the project QUAINT (I774-N23), as well as by the European Commission through project STANCE (317753).

What is Synthesis?

- Specification: **What?**
  - **From:** Graz, Inffeldgasse
  - **To:** Lausanne, 6pm



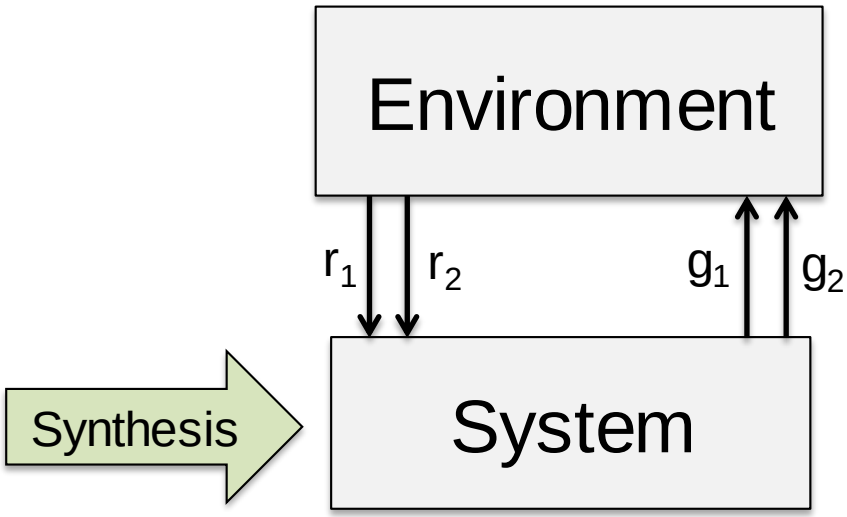
Synthesis

- Implementation: **How?**
  - Walk to Moserhofgasse
  - Tram 6 to Jakominiplatz
    - Buy tram ticket
  - Tram 3 to train station Graz
  - Buy train ticket
  - Train to Salzburg
  - Train to Zürich
  - Train to Launsanne
  - Walk to Lausanne Fon
  - And so on …

Reactive Synthesis

- Specification:
  - Temporal Logic
- Implementation:
  - Reactive system

always($r_1 \rightarrow$ eventually(g_1))
always($r_2 \rightarrow$ eventually(g_2))
never($g_1 \wedge g_2$)



Typical Synthesis Flow

Specification

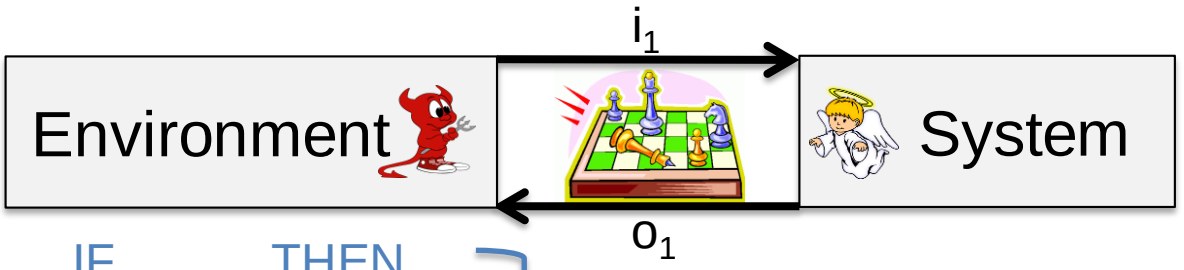
Game

Strategy

Focus of this work

Circuit

always($i_1 \rightarrow$ eventually(o_1))

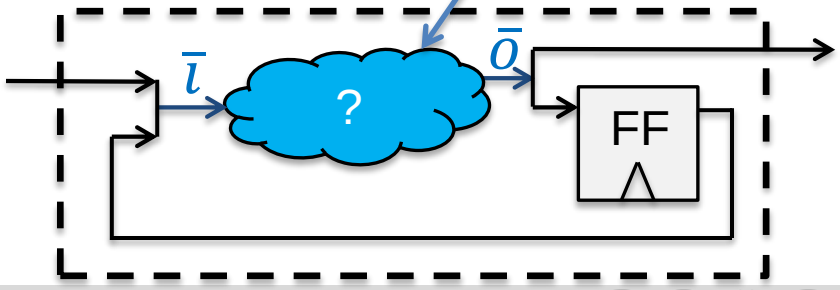


IF THEN

Inputs $\bar{\imath}$		Outputs $\bar{o}$	
i_1	i_2	o_1	o_2
0	0	1	-
0	1	1	1
		0	0

And so on …

Formula $S(\bar{\imath}, \bar{o})$



Challenges

- Scalability
  - Symbolic algorithms
  - Traditionally: BDDs
  - This work: SAT/QBF
- Find small circuits
  - Low number of gates
  - Exploit freedom in $S(\bar{\iota}, \bar{o})$ wisely
- Our work:
  - Comparison of SAT/QBF-based methods
  - Optimizations

Method 1:

QBF Certification

Given:

- $\forall \bar{\imath}: \exists \bar{o}: S(\bar{\imath}, \bar{o})$

Find:

- Skolem function $\bar{o} = f(\bar{\imath})$

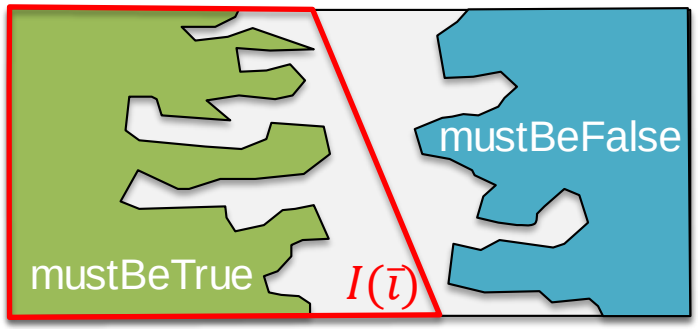
Existing Tool:

- QBFCert [SAT'12]

Method 2: Interpolation [ICCAD'09]

For one output after the other:

- Construct formulas $\mathrm{mustBeTrue}(\vec{\imath})$, $\mathrm{mustBeFalse}(\vec{\imath})$
  - $\mathrm{mustBeTrue}(\vec{\imath}) \wedge \mathrm{mustBeFalse}(\vec{\imath}) = \mathrm{UNSAT}$
- Compute Interpolant $I(\vec{\imath})$
  - $\mathrm{mustBeTrue}(\vec{\imath}) \rightarrow I(\vec{\imath}) \rightarrow \neg\mathrm{mustBeFalse}(\vec{\imath})$



Method 3:

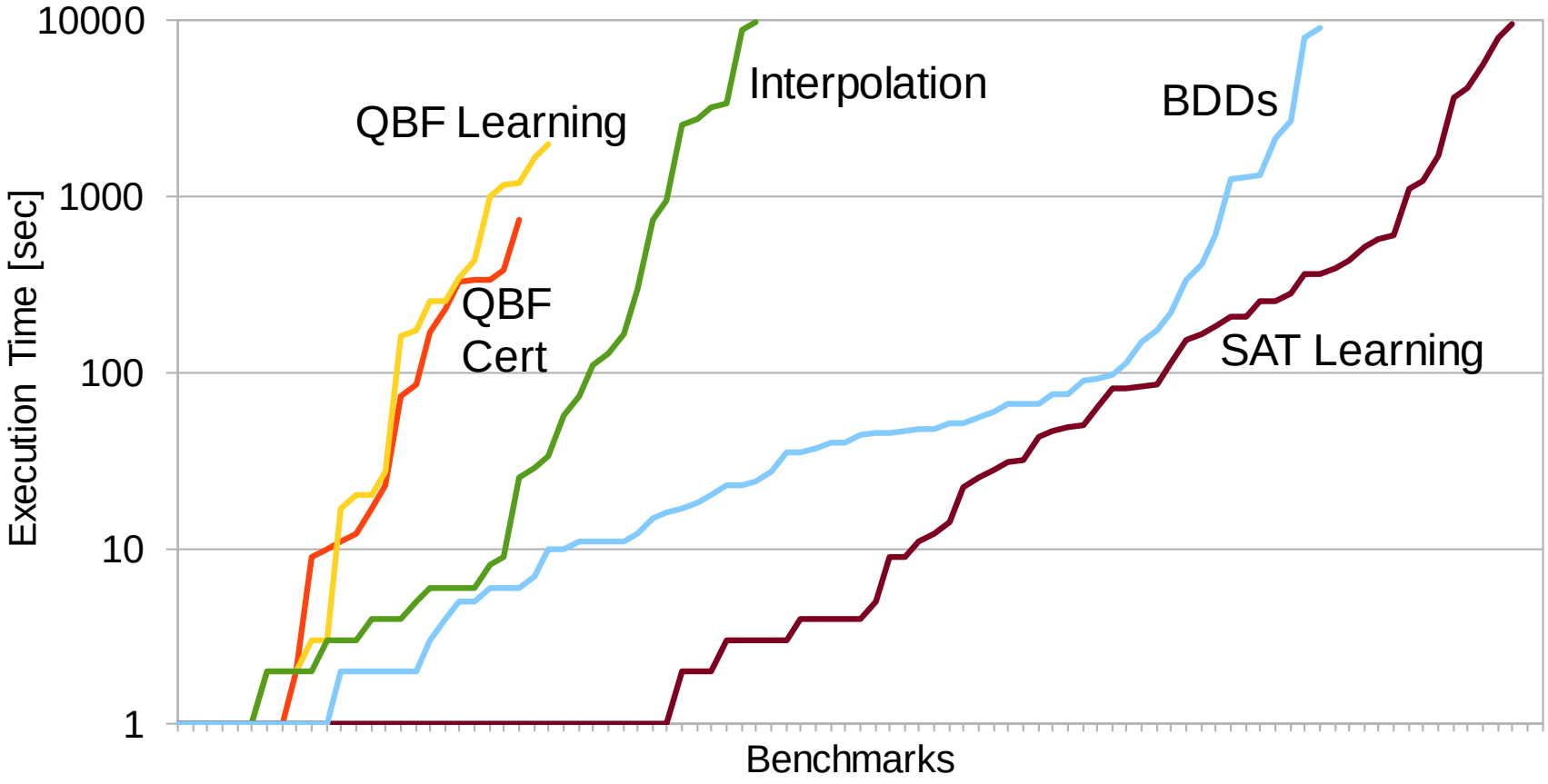
Computational Learning [FMCAD'12]

For one output after the other:

- Construct formulas $\mathrm{mustBeTrue}(\vec{\imath})$, $\mathrm{mustBeFalse}(\vec{\imath})$
  - $\mathrm{mustBeTrue}(\vec{\imath}) \wedge \mathrm{mustBeFalse}(\vec{\imath}) = \mathrm{UNSAT}$
- "Learn" Interpolant $I(\vec{\imath})$
  - Counterexample-guided refinement
  - Many options: SAT or QBF, …

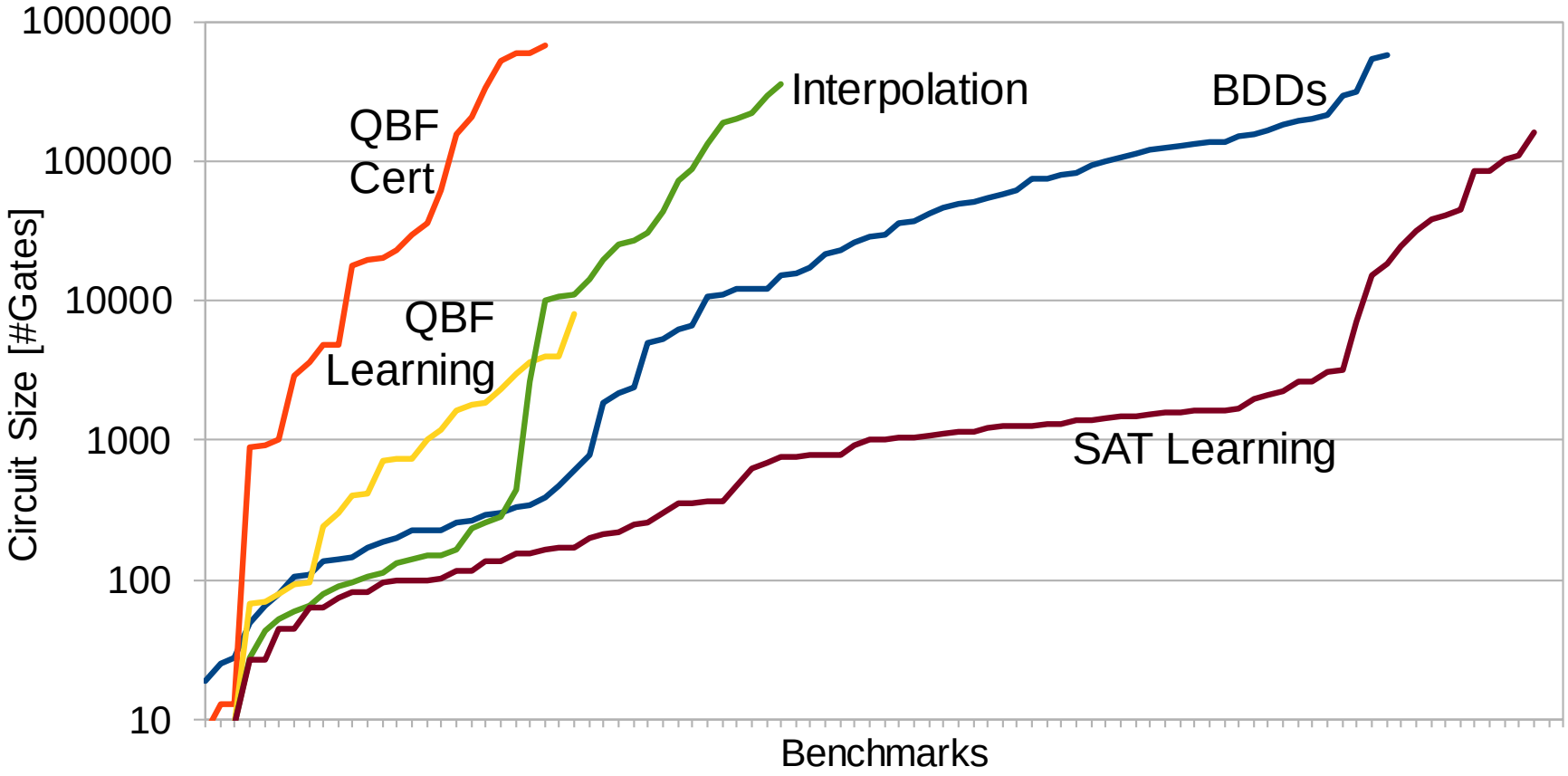
Results: Execution Time

- Cactus Plot



Results: Circuit Size

- Cactus Plot



Conclusions

- SAT-based learning works best
- Execution time and circuit size correlate
- Check out the paper for details
  - Optimizations
  - More results
- Implementation is available:
  - http://www.iaik.tugraz.at/content/research/design_verification/demiurge/

References

[SAT'12] A. Niemetz, M. Preiner, F. Lonsing, M. Seidl, and A. Biere. Resolution- based certificate extraction for QBF. In SAT'12. Springer, 2012.

[ICCAD'09] J.-H. R. Jiang, H.-P. Lin, and W.-L. Hung. Interpolating functions from large boolean relations. In ICCAD'09. IEEE, 2009.

[FMCAD'12] R. Ehlers, R. Könighofer, and G. Hofferek. Symbolically synthesizing small circuits. In FMCAD'12. IEEE, 2012