Solving Q-SAT in bounded space and time by geometrical computation

Denys Duchier¹, Jérôme Durand-Lose², Maxime Senot²

¹ Team Constraint & Machine Learning
² Team Graphs & Algorithms

Laboratoire d’Informatique Fondamentale d’Orléans, Université d’Orléans, Orléans, FRANCE

Partially supported by the ANR AGAPE, ANR-09-BLAN-0159-03.

CiE ’11, Sofia, Bulgaria — 28th June 2011
Solving Q-SAT in bounded space and time by geometrical computation

1. Q-SAT
2. Signal machines
3. Implantation
4. Conclusion
1 Q-SAT

2 Signal machines

3 Implantation

4 Conclusion
Decision problem Q-SAT

Input: a quantified boolean formula $\phi$.
Question: Is $\phi$ true or false?

Example

$$\phi = \exists x_1 \forall x_2 \forall x_3 \ x_1 \land (\neg x_2 \lor x_3)$$

Theorem [Stockmeyer, 1973]

Q-SAT is PSPACE-complete.

On our classical model of computation with the usual notion of complexity.
Solving Q-SAT in bounded space and time by geometrical computation

Q-SAT

Brute force solution

Recursive algorithm

- \( V(\exists x \, \psi) = V(\psi[x \leftarrow false]) \lor V(\psi[x \leftarrow true]) \)
- \( V(\forall x \, \psi) = V(\psi[x \leftarrow false]) \land V(\psi[x \leftarrow true]) \)
- \( V(\beta) = eval(\beta) \) if \( \beta \) is a ground boolean formula.

Polynomial space but exponential time

Example

\[ V(\exists x_1 \, \forall x_2 \, \forall x_3 \, x_1 \land (\neg x_2 \lor x_3)) = \lor \begin{cases} V(\forall x_2 \, \forall x_3 \, false \land (\neg x_2 \lor x_3)) \\ V(\forall x_2 \, \forall x_3 \, true \land (\neg x_2 \lor x_3)) \end{cases} \]

\[ = \lor \begin{cases} \lor \begin{cases} V(\forall x_3 \, false \land (\neg false \lor x_3)) \\ V(\forall x_3 \, false \land (\neg true \lor x_3)) \end{cases} \\ \lor \begin{cases} V(\forall x_3 \, true \land (\neg false \lor x_3)) \\ V(\forall x_3 \, true \land (\neg true \lor x_3)) \end{cases} \end{cases} \]

\[ = \ldots \]
Solving Q-SAT in bounded space and time by geometrical computation

Parallelization scheme

\[ \exists x_1 \forall x_2 \forall x_3 \exists x_1 \forall x_2 \forall x_3 \psi(x_1, x_2, x_3) \text{ where } \psi = x_1 \land (\neg x_2 \lor x_3) \]
Solving Q-SAT in bounded space and time by geometrical computation

Parallelistation scheme

\[ \phi = \exists x_1 \forall x_2 \forall x_3 \; \psi(x_1, x_2, x_3) \text{ where } \psi = x_1 \land (\neg x_2 \lor x_3) \]
Solving Q-SAT in bounded space and time by geometrical computation

Parallelisation scheme

\[ \phi = \exists x_1 \forall x_2 \forall x_3 \psi(x_1, x_2, x_3) \text{ where } \psi = x_1 \land (\neg x_2 \lor x_3) \]
Solving Q-SAT in bounded space and time by geometrical computation

Parallelization scheme

\[
\phi = \exists x_1 \forall x_2 \forall x_3 \psi(x_1, x_2, x_3) \text{ where } \psi = x_1 \land (\neg x_2 \lor x_3)
\]
Solving Q-SAT in bounded space and time by geometrical computation

parallelization scheme

\[ \phi = \exists x_1 \forall x_2 \forall x_3 \psi(x_1, x_2, x_3) \text{ where } \psi = x_1 \land (\neg x_2 \lor x_3) \]
Solving Q-SAT in bounded space and time by geometrical computation

Parallelistation scheme

\[
\phi = \exists x_1 \forall x_2 \forall x_3 \, \psi(x_1, x_2, x_3) \text{ where } \psi = x_1 \land (\neg x_2 \lor x_3)
\]
Solving Q-SAT in bounded space and time by geometrical computation

Q-SAT

Parallelistation scheme

\[ \phi = \exists x_1 \forall x_2 \forall x_3 \psi(x_1, x_2, x_3) \text{ where } \psi = x_1 \land (\neg x_2 \lor x_3) \]
Solving Q-SAT in bounded space and time by geometrical computation

Parallelization scheme

\[ \phi = \exists x_1 \forall x_2 \forall x_3 \psi(x_1, x_2, x_3) \text{ where } \psi = x_1 \land (\neg x_2 \lor x_3) \]
1 Q-SAT
2 Signal machines
3 Implantation
4 Conclusion
“Nice regular drawings”
Solving Q-SAT in bounded space and time by geometrical computation

Signal machines

“Nice regular drawings”
"Nice regular drawings"
“Nice regular drawings”
“Nice regular drawings”
“Nice regular drawings”

Lines: traces of signals
Space-time diagrams of signal machines

Defined by
- bottom: initial configuration
- lines: signals $\leadsto$ meta-signals
- end-points: collisions $\leadsto$ rules
Example: find the middle

Meta-signals (speed)
M (0)

Collision rules
Example: find the middle

Meta-signals (speed)
- \( M (0) \)
- \( \text{div} (3) \)

Collision rules
- \( \{ \text{div, } M \} \rightarrow \{ M, \text{hi, lo} \} \)
- \( \{ \text{lo, } M \} \rightarrow \{ \text{back, } M \} \)
- \( \{ \text{hi, back} \} \rightarrow \{ M \} \)
Solving Q-SAT in bounded space and time by geometrical computation

Signal machines

Example: find the middle

Meta-signals (speed)

- \( M \) (0)
- \( \text{div} \) (3)
- \( \text{hi} \) (1)
- \( \text{lo} \) (3)

Collision rules

\[ \{ \text{div, } M \} \rightarrow \{ M, \text{hi, } \text{lo} \} \]
Example: find the middle

Meta-signals (speed)

- \( M (0) \)
- \( \text{div} (3) \)
- \( \text{hi} (1) \)
- \( \text{lo} (3) \)
- \( \text{back} (-3) \)

Collision rules

\[
\begin{align*}
\{ \text{div, M} \} & \rightarrow \{ M, \text{hi}, \text{lo} \} \\
\{ \text{lo, M} \} & \rightarrow \{ \text{back, M} \}
\end{align*}
\]
Example: find the middle

Meta-signals (speed)
- $M$ (0)
- $\text{div}$ (3)
- $\text{hi}$ (1)
- $\text{lo}$ (3)
- $\text{back}$ (-3)

Collision rules
- $\{ \text{div, M} \} \rightarrow \{ M, \text{hi, lo} \}$
- $\{ \text{lo, M} \} \rightarrow \{ \text{back, M} \}$
- $\{ \text{hi, back} \} \rightarrow \{ M \}$
Known results

Turing computation

- [Durand-Lose, 2011]
### Known results

#### Turing computation
- [Durand-Lose, 2011]

#### Analog computations
- Computable analysis [Weihrauch, 2000]
- [Durand-Lose, 2010]
- Blum, Shub and Smale model [Blum et al., 1989]
- [Durand-Lose, 2008]
Known results

Turing computation
- [Durand-Lose, 2011]

Analog computations
- Computable analysis [Weihrauch, 2000]
  [Durand-Lose, 2010]
- Blum, Shub and Smale model [Blum et al., 1989]
  [Durand-Lose, 2008]

“Black hole” implementation
- Hyper-computation, deciding the Halting problem
  [Durand-Lose, 2009]
1 Q-SAT

2 Signal machines

3 Implantation

4 Conclusion
Solving Q-SAT in bounded space and time by geometrical computation

Implantation

Whole picture

Q-formula compiled into a signal machine

Linear number of meta-signals

Quadratic number of rules
Building the tree with signals
Solving Q-SAT in bounded space and time by geometrical computation

Implantation

Following the tree
Propagating the beam
Evaluating the formula

\[ (x_1 \lor \neg x_2 \land x_3) \]

case:
- \( x_1 \) is true
- \( x_2 \) is false
- \( x_3 \) is true
Collecting the results
Solving Q-SAT in bounded space and time by geometrical computation

Implantation

Whole picture with initialization
Solving Q-SAT in bounded space and time by geometrical computation

Implantation

Generating the formula
1. Q-SAT
2. Signal machines
3. Implantation
4. Conclusion
### Results

- QSAT resolution in bounded space and time
- Quadratic depth
- Exponential width

### Perspectives

- Generic version
- Model of the parallel thesis
- Higher complexity classes
On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines.

Abstract geometrical computation with accumulations: Beyond the Blum, Shub and Smale model.

Abstract geometrical computation 3: Black holes for classical and analog computing.

Abstract geometrical computation 5: embedding computable analysis.

*Nat. Comput.*

Special issue on Unconv. Comp. ’09.

**Durand-Lose, J. (2011).**

Abstract geometrical computation 4: small Turing universal signal machines.


**Weihrauch, K. (2000).**

*Introduction to computable analysis.*