SAT/SMT: a magic way to solve problems

Hans Zantema

Technische Universiteit Eindhoven and Radboud Universiteit Nijmegen

TU/e AI lunch, March 23, 2018
Eight Queens Problem

Can we put 8 queens on the chess board in such a way that no two may hit each other?

Traditional approach: think about how to search for a solution, and write a program finding a solution.

Hans Zantema

SAT/SMT: a magic way to solve problems
Eight Queens Problem

Can we put 8 queens on the chess board in such a way that no two may hit each other?

Traditional approach: think about how to search for a solution, and write a program finding a solution.

Hans Zantema

SAT/SMT: a magic way to solve problems
Eight Queens Problem

Can we put 8 queens on the chess board in such a way that no two may hit each other?
Eight Queens Problem

Can we put 8 queens on the chess board in such a way that no two may hit each other?

Traditional approach: think about how to search for a solution, and write a program finding a solution.
AI approach: only specify the problem, and let a general tool search for a solution

Traditional AI already used Prolog as a programming language, in which not the steps of the program are given but only constraints are specified.

Currently we use SAT/SMT:

- **SAT** = satisfiability:
  - Given a formula composed from Boolean variables and operations \( \neg \) (not), \( \lor \) (or), \( \land \) (and), can we give values to the variables such that the formula yields true?

  - Example: \((p \lor q) \land (\neg p \lor \neg q) \land (\neg p \lor q)\) is satisfiable: by choosing \(p = \text{false}\) and \(q = \text{true}\) the formula yields true.
AI approach: only specify the problem, and let a general tool search for a solution

Traditional AI already used Prolog as a programming language, in which not the steps of the program are given but only constraints are specified

SAT/SMT = satisfiability: Given a formula composed from Boolean variables and operations ¬ (not), ∨ (or), ∧ (and), can we give values to the variables such that the formula yields true?

Example (p ∨ q) ∧ (¬ p ∨ ¬ q) ∧ (¬ p ∨ q) is satisfiable: by choosing p = false and q = true the formula yields true
AI approach: only specify the problem, and let a general tool search for a solution

Traditional AI already used Prolog as a programming language, in which not the steps of the program are given but only constraints are specified

Currently we use SAT/SMT
AI approach: only specify the problem, and let a general tool search for a solution

Traditional AI already used Prolog as a programming language, in which not the steps of the program are given but only constraints are specified

Currently we use SAT/SMT

SAT = satisfiability:
Given a formula composed from Boolean variables and operations ¬ (not), ∨ (or), ∧ (and), can we give values to the variables such that the formula yields true?

Example (p ∨ q) ∧ (¬p ∨ ¬q) ∧ (¬p ∨ q) is satisfiable: by choosing p = false and q = true the formula yields true
AI approach: only specify the problem, and let a general tool search for a solution

Traditional AI already used *Prolog* as a programming language, in which not the steps of the program are given but only constraints are specified

Currently we use *SAT/SMT*

**SAT = satisfiability:**
Given a formula composed from Boolean variables and operations ¬ (not), ∨ (or), ∧ (and), can we give values to the variables such that the formula yields true?

*Example*

\[(p \lor q) \land (\neg p \lor \neg q) \land (\neg p \lor q)\] is satisfiable:
AI approach: only specify the problem, and let a general tool search for a solution

Traditional AI already used \textit{Prolog} as a programming language, in which not the steps of the program are given but only constraints are specified

Currently we use \textit{SAT/SMT}

\textit{SAT} = \textit{satisfiability}:
Given a formula composed from Boolean variables and operations \(\neg\) (not), \(\lor\) (or), \(\land\) (and), can we give values to the variables such that the formula yields true?

\textit{Example}
\((p \lor q) \land (\neg p \lor \neg q) \land (\neg p \lor q)\) is satisfiable:

by choosing \(p = \text{false}\) and \(q = \text{true}\) the formula yields true
SAT is *NP-hard*, the main standard of hard problems in computer science.
SAT is *NP-hard*, the main standard of hard problems in computer science.

In fact it was the starting point of all theory on NP-hardness in 1970.
SAT is *NP-hard*, the main standard of hard problems in computer science.

In fact it was the starting point of all theory on NP-hardness in 1970.

Indeed there are small formulas for which SAT is hard to establish.
SAT is *NP-hard*, the main standard of hard problems in computer science.

In fact it was the starting point of all theory on NP-hardness in 1970.

Indeed there are small formulas for which SAT is hard to establish.

But there are many big formulas expressing practical problems from a wide range of application areas that can be solved by current SAT solvers.
SAT is *NP-hard*, the main standard of hard problems in computer science.

In fact it was the starting point of all theory on NP-hardness in 1970.

Indeed there are small formulas for which SAT is hard to establish.

But there are many big formulas expressing practical problems from a wide range of application areas that can be solved by current SAT solvers.

100 queens on \(100 \times 100\) chess board yields 50Mb formula on 10,000 variables, solved within seconds.
SMT solving

A wide range of theories can be used, but the most important is:
linear inequalities
Apart from boolean variables, now also integer or real variables
may be used, and linear inequalities on them like
\[2x + 3y - z \geq 17\]
Step further than linear programming, in which the constraints are
a conjunction of linear inequalities: in SMT also the other boolean
operators \(\neg\), \(\lor\) may be used

Hans Zantema

SAT/SMT: a magic way to solve problems
SMT solving

**SMT**: Satisfiability Modulo Theories

Apart from boolean variables, now also integer or real variables may be used, and linear inequalities on them like

\[2x + 3y - z \geq 17\]

Step further than linear programming, in which the constraints are a conjunction of linear inequalities: in SMT also the other boolean operators \(\neg, \lor\) may be used.
SMT solving

SMT: Satisfiability Modulo Theories

A wide range of theories can be used, but the most important is: *linear inequalities*
SMT : Satisfiability Modulo Theories

A wide range of theories can be used, but the most important is: *linear inequalities*

Apart from boolean variables, now also integer or real variables may be used, and linear inequalities on them like

\[ 2x + 3y - z \geq 17 \]
SMT solving

**SMT** : Satisfiability Modulo Theories

A wide range of theories can be used, but the most important is: *linear inequalities*

Apart from boolean variables, now also integer or real variables may be used, and linear inequalities on them like

\[ 2x + 3y - z \geq 17 \]

Step further than *linear programming*, in which the constraints are a conjunction of linear inequalities: in SMT also the other boolean operators \( \neg, \lor \) may be used
Example: rectangle fitting

Given: size of a big rectangle, sizes of several small rectangles
Problem: fit the small rectangles inside the big one without overlap

Several applications:
- design a chip from rectangular components
- poster printing

Hans Zantema  SAT/SMT: a magic way to solve problems
Example: rectangle fitting

Given: size of a big rectangle, sizes of several small rectangles
Example: rectangle fitting

Given: size of a big rectangle, sizes of several small rectangles
Problem: fit the small rectangles inside the big one without overlap
Example: rectangle fitting

Given: size of a big rectangle, sizes of several small rectangles
Problem: fit the small rectangles inside the big one without overlap
Example: rectangle fitting

Given: size of a big rectangle, sizes of several small rectangles
Problem: fit the small rectangles inside the big one without overlap

Several applications:
Example: rectangle fitting

Given: size of a big rectangle, sizes of several small rectangles
Problem: fit the small rectangles inside the big one without overlap

Several applications:

- design a chip from rectangular components
Example: rectangle fitting

Given: size of a big rectangle, sizes of several small rectangles
Problem: fit the small rectangles inside the big one without overlap

Several applications:

- design a chip from rectangular components
- poster printing
Main applications

Planning / scheduling
Program verification / bounded model checking
For every program variable $a$, introduce variable $a_i$ for value of $a$ after $i$ steps.
Describe meaning of program in these variables, and add negation of desired property.
If this formula is unsatisfiable, then the program satisfies the desired property.

Find solutions of mathematical / combinatorial problems...

Hans Zantema

SAT/SMT: a magic way to solve problems
Main applications

- Planning / scheduling

For every program variable \( a \) introduce variable \( a_i \) for value of \( a \) after \( i \) steps. Describe meaning of program in these variables, and add negation of desired property. If this formula is unsatisfiable, then the program satisfies the desired property.

Find solutions of mathematical / combinatorial problems.

SAT/SMT: a magic way to solve problems
Main applications

- Planning / scheduling
- Program verification / bounded model checking

For every program variable $a$, introduce variable $a_i$ for value of $a$ after $i$ steps. Describe meaning of program in these variables, and add negation of desired property. If this formula is unsatisfiable, then the program satisfies the desired property.

Hans Zantema

SAT/SMT: a magic way to solve problems
Main applications

- Planning / scheduling
- Program verification / bounded model checking

For every program variable $a$ introduce variable $a_i$ for value of $a$ after $i$ steps

Describe meaning of program in these variables, and add negation of desired property. If this formula is unsatisfiable, then the program satisfies the desired property.
Main applications

- Planning / scheduling

- Program verification / bounded model checking

  For every program variable \( a \) introduce variable \( a_i \) for value of \( a \) after \( i \) steps

  Describe meaning of program in these variables, and add negation of desired property

Hans Zantema
SAT/SMT: a magic way to solve problems
Main applications

- Planning / scheduling
- Program verification / bounded model checking

For every program variable $a$ introduce variable $a_i$ for value of $a$ after $i$ steps.

Describe meaning of program in these variables, and add negation of desired property.

If this formula is unsatisfiable, then the program satisfies the desired property.
Main applications

- Planning / scheduling
- Program verification / bounded model checking

For every program variable $a$ introduce variable $a_i$ for value of $a$ after $i$ steps.

Describe meaning of program in these variables, and add negation of desired property.

If this formula is unsatisfiable, then the program satisfies the desired property.

- find solutions of mathematical / combinatorial problems
Main applications

- Planning / scheduling
- Program verification / bounded model checking

For every program variable $a$ introduce variable $a_i$ for value of $a$ after $i$ steps

Describe meaning of program in these variables, and add negation of desired property

If this formula is unsatisfiable, then the program satisfies the desired property

- find solutions of mathematical / combinatorial problems
- ......
Conclusions

SAT/SMT is a successful approach to solve problems in a wide range of areas, as long as they can be expressed as finding solutions for a given set of constraints. Human intelligence for heuristics for searching for solutions is replaced by combinations of computer power and clever heuristics exploited in SAT/SMT solvers. In contrast to several other approaches (e.g., genetic algorithms), by SAT/SMT you not only find solutions, but also may prove that they do not exist. The latter is exploited for program correctness. Free available SAT/SMT solvers: Z3, Yices, CVC4, . . .

Course: Automated Reasoning and mooc on this topic.
SAT/SMT is a successful approach to solve problems in a wide range of areas, as long as they can be expressed as finding solutions for a given set of constraints.
Conclusions

- SAT/SMT is a successful approach to solve problems in a wide range of areas, as long as they can be expressed as finding solutions for a given set of constraints.

- Human intelligence for heuristics for searching for solutions is replaced by combinations of computer power and clever heuristics exploited in SAT/SMT solvers.

In contrast to several other approaches (e.g., genetic algorithms), by SAT/SMT you not only find solutions, but also may prove that they do not exist. The latter is exploited for program correctness.

Free available SAT/SMT solvers: Z3, Yices, CVC4, ...
Conclusions

- SAT/SMT is a successful approach to solve problems in a wide range of areas, as long as they can be expressed as finding solutions for a given set of constraints.

- Human intelligence for heuristics for searching for solutions is replaced by combinations of computer power and clever heuristics exploited in SAT/SMT solvers.

- In contrast to several other approaches (e.g., genetic algorithms), by SAT/SMT you not only find solutions, but also may prove that they do not exist.
Conclusions

- SAT/SMT is a successful approach to solve problems in a wide range of areas, as long as they can be expressed as finding solutions for a given set of constraints.

- Human intelligence for heuristics for searching for solutions is replaced by combinations of computer power and clever heuristics exploited in SAT/SMT solvers.

- In contrast to several other approaches (e.g., genetic algorithms), by SAT/SMT you not only find solutions, but also may prove that they do not exist.

- The latter is exploited for program correctness.
Conclusions

- SAT/SMT is a successful approach to solve problems in a wide range of areas, as long as they can be expressed as finding solutions for a given set of constraints.

- Human intelligence for heuristics for searching for solutions is replaced by combinations of computer power and clever heuristics exploited in SAT/SMT solvers.

- In contrast to several other approaches (e.g., genetic algorithms), by SAT/SMT you not only find solutions, but also may prove that they do not exist.

- The latter is exploited for program correctness.

- Free available SAT/SMT solvers: Z3, Yices, CVC4, ...
Conclusions

- SAT/SMT is a successful approach to solve problems in a wide range of areas, as long as they can be expressed as finding solutions for a given set of constraints.

- Human intelligence for heuristics for searching for solutions is replaced by combinations of computer power and clever heuristics exploited in SAT/SMT solvers.

- In contrast to several other approaches (e.g., genetic algorithms), by SAT/SMT you not only find solutions, but also may prove that they do not exist.

- The latter is exploited for program correctness.

- Free available SAT/SMT solvers: Z3, Yices, CVC4, . . .

- Course *Automated Reasoning* and mooc on this topic.