

Equational Reasoning and Constraint Solving

Hans Zantema

Department of Computer Science, TU Eindhoven, and
Computing Science Department, Radboud University Nijmegen
The Netherlands
email: h.zantema@tue.nl

Seminar FSA, September 15, 2011

Constraint solving

SAT solving: given a propositional formula in boolean variables, determine whether it is *satisfiable*, that is, yields true for some valuation of the variables

SAT solving: given a propositional formula in boolean variables, determine whether it is *satisfiable*, that is, yields true for some valuation of the variables

Surprise: a wide range of problems can be expressed as a SAT problem and solved by current SAT solvers, ranging from sudoku to hardware verification

Constraint solving

SAT solving: given a propositional formula in boolean variables, determine whether it is *satisfiable*, that is, yields true for some valuation of the variables

Surprise: a wide range of problems can be expressed as a SAT problem and solved by current SAT solvers, ranging from sudoku to hardware verification

SMT solving: *satisfiability modulo theories*

SAT solving: given a propositional formula in boolean variables, determine whether it is *satisfiable*, that is, yields true for some valuation of the variables

Surprise: a wide range of problems can be expressed as a SAT problem and solved by current SAT solvers, ranging from sudoku to hardware verification

SMT solving: *satisfiability modulo theories*

Now the atoms in the propositional formula are not only boolean variables, but also can be (linear) (in)equalities in integer or real variables

SAT solving: given a propositional formula in boolean variables, determine whether it is *satisfiable*, that is, yields true for some valuation of the variables

Surprise: a wide range of problems can be expressed as a SAT problem and solved by current SAT solvers, ranging from sudoku to hardware verification

SMT solving: *satisfiability modulo theories*

Now the atoms in the propositional formula are not only boolean variables, but also can be (linear) (in)equalities in integer or real variables

Even more problems can be expressed

This floor planning problem occurs in practice at several places, e.g., placement in chip design (NXP), to be extended by routing, that is, chip components are not only placed on a position, but also have pins that should be connected

This floor planning problem occurs in practice at several places, e.g., placement in chip design (NXP), to be extended by routing, that is, chip components are not only placed on a position, but also have pins that should be connected

Last year, Peter van Otterdijk did this seminar on the placement problem, followed by a master's project at NXP where he made a preliminary implementation for routing based on SMT

This floor planning problem occurs in practice at several places, e.g., placement in chip design (NXP), to be extended by routing, that is, chip components are not only placed on a position, but also have pins that should be connected

Last year, Peter van Otterdijk did this seminar on the placement problem, followed by a master's project at NXP where he made a preliminary implementation for routing based on SMT

Variants and extensions still worthwhile to be investigated

Several other practical problems can be expressed in SMT, like scheduling (jobs, educational programs, packages in trucs, . . .)

Several other practical problems can be expressed in SMT, like scheduling (jobs, educational programs, packages in trucs, . . .)

Of particular interest: program verification

Several other practical problems can be expressed in SMT, like scheduling (jobs, educational programs, packages in trucs, . . .)

Of particular interest: program verification

For instance at Microsoft Research in Cambridge, the team of Byron Cook developed the tool Terminator, by which thousands of external drivers are checked for termination, mainly based on SMT solving

Several other practical problems can be expressed in SMT, like scheduling (jobs, educational programs, packages in trucs, . . .)

Of particular interest: program verification

For instance at Microsoft Research in Cambridge, the team of Byron Cook developed the tool Terminator, by which thousands of external drivers are checked for termination, mainly based on SMT solving

The tool itself is secret, but many underlying ideas are public

Several other practical problems can be expressed in SMT, like scheduling (jobs, educational programs, packages in trucs, . . .)

Of particular interest: program verification

For instance at Microsoft Research in Cambridge, the team of Byron Cook developed the tool Terminator, by which thousands of external drivers are checked for termination, mainly based on SMT solving

The tool itself is secret, but many underlying ideas are public

Investigating these and experiment with some small examples would be a nice topic for the seminar

Equational reasoning / term rewriting

Lot of work has been done in proving termination automatically, often using SAT solving

Lot of work has been done in proving termination automatically, often using SAT solving

That is, given a set of rewrite rules, prove that it does not allow infinite computations

Lot of work has been done in proving termination automatically, often using SAT solving

That is, given a set of rewrite rules, prove that it does not allow infinite computations

Recent master's project (Evans Kaijage): investigate criteria for which the number of rewrite steps for reaching a result is independent of chosen strategy

Lot of work has been done in proving termination automatically, often using SAT solving

That is, given a set of rewrite rules, prove that it does not allow infinite computations

Recent master's project (Evans Kaijage): investigate criteria for which the number of rewrite steps for reaching a result is independent of chosen strategy

Interesting term rewrite system: standard arithmetic on binary numbers, e.g., as implemented in mCRL2

Lot of work has been done in proving termination automatically, often using SAT solving

That is, given a set of rewrite rules, prove that it does not allow infinite computations

Recent master's project (Evans Kaijage): investigate criteria for which the number of rewrite steps for reaching a result is independent of chosen strategy

Interesting term rewrite system: standard arithmetic on binary numbers, e.g., as implemented in mCRL2

This term rewrite system allows several variants, of which investigation of properties would be a nice seminar topic

Last year's seminar student Arjen Meurers did his master's project on developing an SMT based tool for negative results for the *word problem*, that is, for proving that some equality cannot be derived from a given set of equations

Last year's seminar student Arjen Meurers did his master's project on developing an SMT based tool for negative results for the *word problem*, that is, for proving that some equality cannot be derived from a given set of equations

Example:

Last year's seminar student Arjen Meurers did his master's project on developing an SMT based tool for negative results for the *word problem*, that is, for proving that some equality cannot be derived from a given set of equations

Example:

Can the equation $s(s(s(0))) = s(s(0))$ be derived from the equations

$$\begin{aligned}x + 0 &= x \\s(x) + y &= s(x + y) \\x + y &= y + x\end{aligned}$$

Last year's seminar student Arjen Meurers did his master's project on developing an SMT based tool for negative results for the *word problem*, that is, for proving that some equality cannot be derived from a given set of equations

Example:

Can the equation $s(s(s(0))) = s(s(0))$ be derived from the equations

$$\begin{aligned}x + 0 &= x \\s(x) + y &= s(x + y) \\x + y &= y + x\end{aligned}$$

NO: interpret 0, s, + by its usual meaning, then all equations hold, but $s(s(s(0))) = s(s(0))$ does not hold

Stream specifications

Stream specifications

The Fibonacci stream $\text{FIB} = 010010100100101 \dots$ satisfies

$$\text{FIB} = f(\text{FIB})$$

$$f(0 : \sigma) = 0 : 1 : f(\sigma)$$

$$f(1 : \sigma) = 0 : f(\sigma)$$

Stream specifications

The Fibonacci stream $\text{FIB} = 010010100100101 \dots$ satisfies

$$\text{FIB} = f(\text{FIB})$$

$$f(0 : \sigma) = 0 : 1 : f(\sigma)$$

$$f(1 : \sigma) = 0 : f(\sigma)$$

Is FIB the only stream satisfying these equations?

Stream specifications

The Fibonacci stream $\text{FIB} = 010010100100101 \dots$ satisfies

$$\text{FIB} = f(\text{FIB})$$

$$f(0 : \sigma) = 0 : 1 : f(\sigma)$$

$$f(1 : \sigma) = 0 : f(\sigma)$$

Is FIB the only stream satisfying these equations?

Do two sets of equations define the same stream?

Stream specifications

The Fibonacci stream $\text{FIB} = 010010100100101 \dots$ satisfies

$$\text{FIB} = f(\text{FIB})$$

$$f(0 : \sigma) = 0 : 1 : f(\sigma)$$

$$f(1 : \sigma) = 0 : f(\sigma)$$

Is FIB the only stream satisfying these equations?

Do two sets of equations define the same stream?

For this kind of questions several techniques have been developed to do this fully automatically, some of which using termination, or an implementation for the word problem

Stream specifications

The Fibonacci stream $\text{FIB} = 010010100100101 \dots$ satisfies

$$\text{FIB} = f(\text{FIB})$$

$$f(0 : \sigma) = 0 : 1 : f(\sigma)$$

$$f(1 : \sigma) = 0 : f(\sigma)$$

Is FIB the only stream satisfying these equations?

Do two sets of equations define the same stream?

For this kind of questions several techniques have been developed to do this fully automatically, some of which using termination, or an implementation for the word problem

Last year, Frank Staals did his seminar project on treating these issues in Coq

Stream specifications

The Fibonacci stream $\text{FIB} = 010010100100101 \dots$ satisfies

$$\text{FIB} = f(\text{FIB})$$

$$f(0 : \sigma) = 0 : 1 : f(\sigma)$$

$$f(1 : \sigma) = 0 : f(\sigma)$$

Is FIB the only stream satisfying these equations?

Do two sets of equations define the same stream?

For this kind of questions several techniques have been developed to do this fully automatically, some of which using termination, or an implementation for the word problem

Last year, Frank Staals did his seminar project on treating these issues in Coq

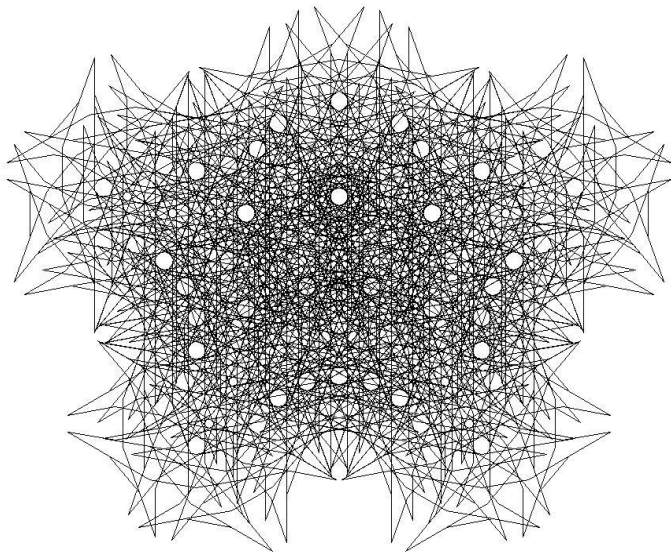
Many remaining questions on streams suitable for seminar/master project

Visualization of Fibonacci stream

$N =$
18.000

$\alpha = 20^\circ$

$\beta = 160^\circ$



Summary, suitable topics

Summary, suitable topics

- Investigate applications of constraint solving, in particular SMT solving for e.g.

Summary, suitable topics

- Investigate applications of constraint solving, in particular SMT solving for e.g.
 - Design with physical/geometrical aspects

- Investigate applications of constraint solving, in particular SMT solving for e.g.
 - Design with physical/geometrical aspects
 - Program verification

- Investigate applications of constraint solving, in particular SMT solving for e.g.
 - Design with physical/geometrical aspects
 - Program verification
- Equational reasoning / term rewriting

- Investigate applications of constraint solving, in particular SMT solving for e.g.
 - Design with physical/geometrical aspects
 - Program verification
- Equational reasoning / term rewriting
- Investigate stream definitions / properties