SAT and Termination

Nao Hirokawa

Japan Advanced Institute of Science and Technology

Sudoku Puzzle

given 9×9-grid like

	1	8				7		
			3			2		
	7							
				7	1			
6							4	
3								
4			5					3
	2			8				
							6	

fill out numbers from 1 to 9 that each number appears exactly once in each {row, column, 3×3-subgrid}

Sudoku Puzzle

given 9×9-grid like

	1	8				7		
			3			2		
	7							
				7	1			
6							4	
3								
4			5					3
	2			8				
							6	

fill out numbers from 1 to 9 that each number appears exactly once in each $\{$row, column, 3×3-subgrid$\}$

how to solve?

Sudoku Puzzle

given 9×9-grid like

	1	8				7		
			3			2		
	7							
				7	1			
6							4	
3								
4			5					3
	2			8				
							6	

fill out numbers from 1 to 9 that each number appears exactly once in each {row, column, 3×3-subgrid}

how to solve? — difficult?

Sudoku Puzzle

given 9×9-grid like

	1	**8**				**7**		
			3			**2**		
	7							
				7	**1**			
6							**4**	
3								
4			**5**					**3**
	2			**8**				
							6	

fill out numbers from 1 to 9 that each number appears exactly once in each $\{$row, column, 3×3-subgrid$\}$

how to solve? — difficult? — NP-complete

Solving Sudoku

x_{11}	**1**	**8**				**7**		
x_{21}	x_{22}	x_{23}	**3**			**2**		
x_{31}	**7**	x_{33}						
				7	**1**			
6							**4**	
3								
4			**5**					**3**
	2			**8**				
							6	

Solving Sudoku

x_{11}	**1**	**8**				**7**		
x_{21}	x_{22}	x_{23}	**3**			**2**		
x_{31}	**7**	x_{33}						
				7	**1**			
6							**4**	
3								
4			**5**					**3**
	2			**8**				
							6	

Lemma

$x_{33} = 3$

Proof

- $\{x_{11}, x_{21}, x_{22}, x_{23}, x_{31}, x_{33}\} = \{2, 3, 4, 5, 6, 9\}$ by subgrid constraint
- $\{x_{11}, x_{21}, x_{22}, x_{23}, x_{31}\} = \{2, 4, 5, 6, 9\}$ by row & column constraint

$\square$

how about x_{26} ?

Solving Sudoku

	1	**8**				**7**		
x_{21}	x_{22}	x_{23}	**3**	x_{25}	x_{26}	**2**	x_{28}	x_{29}
	7	**3**						
				7	**1**			
6							**4**	
3								
4			**5**					**3**
	2			**8**				
							6	

Solving Sudoku

	1	8				7		
x_{21}	x_{22}	x_{23}	3	x_{25}	x_{26}	2	x_{28}	x_{29}
	7	3						
				7	1			
6							4	
3								
4			5					3
	2			8				
							6	

LEMMA

$x_{26} = 7$

Solving Sudoku

	1	**8**				**7**		
x_{21}	x_{22}	x_{23}	**3**	x_{25}	x_{26}	**2**	x_{28}	x_{29}
	7	**3**						
				7	**1**			
6							**4**	
3								
4			**5**					**3**
	2			**8**				
							6	

Lemma

$x_{26} = 7$

Proof

- $\{x_{21}, x_{22}, x_{23}, x_{25}, x_{26}, x_{28}, x_{29}\} = \{1, 4, 5, 6, 7, 8, 9\}$
  by row constraint
- $7 \notin \{x_{21}, x_{22}, x_{23}, x_{25}, x_{28}, x_{29}\}$ by column & subgrid constraint

$\square$

Overview

- SAT

Overview

- SAT

- encoding techniques

Overview

- SAT
- encoding techniques
- termination analysis

SAT

DEFINITION

- syntax

$$\begin{array}{rcll} \ell & ::= & x \mid \neg x & \text{literal} \\ C & ::= & \ell_1 \vee \cdots \vee \ell_n & \text{clause} \\ \phi & ::= & C_1 \wedge \cdots \wedge C_n & \text{CNF} \end{array}$$

SAT

DEFINITION

- syntax

$$
\begin{aligned}
\ell &::= x \mid \neg x && \text{literal} \\
C &::= \ell_1 \vee \cdots \vee \ell_n && \text{clause} \\
\phi &::= C_1 \wedge \cdots \wedge C_n && \text{CNF}
\end{aligned}
$$

- SAT problem is decision problem of scheme:

instance:	CNF ϕ
question:	is ϕ satisfiable ?

SAT

DEFINITION

- syntax

$$
\begin{aligned}
\ell &::= x \mid \neg x && \text{literal} \\
C &::= \ell_1 \vee \cdots \vee \ell_n && \text{clause} \\
\phi &::= C_1 \wedge \cdots \wedge C_n && \text{CNF}
\end{aligned}
$$

- SAT problem is decision problem of scheme:

instance: CNF ϕ

question: is ϕ satisfiable ?

THEOREM Cook and Levin

SAT is NP-complete

Quiz

is next CNF ϕ satisfiable?

$$\bigwedge \left\{ \begin{array}{llll} & x_1 \vee \neg x_2, & & \\ \neg x_1 \vee & x_2 \vee & x_3, & \\ \neg x_2 \vee & x_3, & & \\ \neg x_3 & & & \end{array} \right\}$$

Quiz

is next CNF ϕ satisfiable?

$$\bigwedge \left\{ \begin{array}{llll} x_1 \vee & \neg x_2, & & \\ \neg x_1 \vee & x_2 \vee & x_3, & \\ \neg x_2 \vee & x_3, & & \\ \neg x_3 & & & \end{array} \right\}$$

ϕ is satisfiable

Quiz

is next CNF ϕ satisfiable?

$$\bigwedge \left\{ \begin{array}{llll} & x_1 \vee \neg x_2, & & \\ \neg x_1 \vee & x_2 \vee & x_3, & \\ \neg x_2 \vee & x_3, & & \\ \neg x_3 & & & \end{array} \right\}$$

ϕ is satisfiable because of next satisfiable assignment

$$x_1 \mapsto F \qquad x_2 \mapsto F \qquad x_3 \mapsto F$$

Quiz

is next CNF ϕ satisfiable?

$$\bigwedge \left\{ \begin{array}{l} \quad x_1 \vee \neg x_2, \\ \neg x_1 \vee \neg x_2 \vee \quad x_3, \\ \neg x_2 \vee \quad x_3, \\ \neg x_3 \end{array} \right\}$$

Quiz

is next CNF ϕ satisfiable?

$$\bigwedge \left\{ \begin{array}{llll} x_1 \vee \neg x_2, & & \\ \neg x_1 \vee \neg x_2 \vee & x_3, & \\ \neg x_2 \vee & x_3, & \\ \neg x_3 & & \end{array} \right\}$$

ϕ is unsatisfiable. why?

Quiz

is next CNF ϕ satisfiable?

$$\bigwedge \left\{ \begin{array}{llll} x_1 \vee \neg x_2, & & \\ \neg x_1 \vee \neg x_2 \vee & x_3, & \\ \neg x_2 \vee \quad x_3, & & \\ \neg x_3 & & \end{array} \right\}$$

ϕ is unsatisfiable. why?

because

- exhaustive search shows that any assignment is unsatisfiable, or

Quiz

is next CNF ϕ satisfiable?

$$\bigwedge \left\{ \begin{array}{llll} x_1 \vee \neg x_2, & & \\ \neg x_1 \vee \neg x_2 \vee & x_3, & \\ \neg x_2 \vee \quad x_3, & & \\ \neg x_3 & & \end{array} \right\}$$

ϕ is unsatisfiable.   why?

because

- exhaustive search shows that any assignment is unsatisfiable, or
- deduction derives ϕ is $\perp$

SAT Solver and DIMACS Format

is $\bigwedge \left\{ \begin{array}{l} x_1 \vee \neg x_2, \\ \neg x_1 \vee x_2 \vee \neg x_3, \\ x_2 \vee \neg x_3, \\ \neg x_3 \end{array} \right\}$ satisfiable ?

SAT Solver and DIMACS Format

is $\bigwedge \left\{ \begin{array}{l} \quad x_1 \vee \neg x_2, \\ \neg x_1 \vee \quad x_2 \vee \neg x_3, \\ \quad x_2 \vee \neg x_3, \\ \neg x_3 \end{array} \right\}$ satisfiable ?

```
$ cat a.cnf
p cnf 3 #variables 4 #clauses
 1 -2  0
-1  2 -3  0
 3 -2  0
-3  0
```

```
$ minisat a.cnf a.ans
$ cat a.ans
SAT
1 -2 -3 0
```

Sudoku Constraints

	1	8				7		
			3			2		
	7							
				7	1			
6							4	
3								
4			5					3
	2			8				
							6	

constraints

- each cell is number from 1 to 9
- each number appears at most once in each row
- each number appears at most once in each column
- each number appears at most once in each 3×3 subgrid

Sudoku Constraints

	1	8				7		
			3			2		
	7							
				7	1			
6							4	
3								
4			5					3
	2			8				
							6	

constraints

- each cell is number from 1 to 9
- each number appears at most once in each row
- each number appears at most once in each column
- each number appears at most once in each 3×3 subgrid

☞ use $s_{111}, \ldots, s_{999}$ with $s_{ijk} = T \iff x_{ij} = k$

Sudoku Constraints in Boolean

- each cell is number from 1 to 9

$$\bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} (s_{ij1} \vee \cdots \vee s_{ij9})$$

Sudoku Constraints in Boolean

- each cell is number from 1 to 9

$$\bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} (s_{ij1} \vee \cdots \vee s_{ij9})$$

- each number appears at most once in each row

$$\bigwedge_{j=1}^{9} \bigwedge_{k=1}^{9} \bigwedge_{i=1}^{9} \bigwedge_{i'=i+1}^{9} (\neg s_{ijk} \vee \neg s_{i'jk})$$

Sudoku Constraints in Boolean

- each cell is number from 1 to 9

$$\bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} (s_{ij1} \vee \cdots \vee s_{ij9})$$

- each number appears at most once in each row

$$\bigwedge_{j=1}^{9} \bigwedge_{k=1}^{9} \bigwedge_{i=1}^{9} \bigwedge_{i'=i+1}^{9} (\neg s_{ijk} \vee \neg s_{i'jk})$$

- each number appears at most once in each column

$$\bigwedge_{i=1}^{9} \bigwedge_{k=1}^{9} \bigwedge_{j=1}^{9} \bigwedge_{j'=j+1}^{9} (\neg s_{ijk} \vee \neg s_{ij'k})$$

Sudoku Constraints in Boolean

- each cell is number from 1 to 9

$$\bigwedge_{i=1}^{9}\bigwedge_{j=1}^{9}(s_{ij1} \vee \cdots \vee s_{ij9})$$

- each number appears at most once in each row

$$\bigwedge_{j=1}^{9}\bigwedge_{k=1}^{9}\bigwedge_{i=1}^{9}\bigwedge_{i'=i+1}^{9}(\neg s_{ijk} \vee \neg s_{i'jk})$$

- each number appears at most once in each column

$$\bigwedge_{i=1}^{9}\bigwedge_{k=1}^{9}\bigwedge_{j=1}^{9}\bigwedge_{j'=j+1}^{9}(\neg s_{ijk} \vee \neg s_{ij'k})$$

- each number appears at most once in each 3×3 subgrid

$$\bigwedge_{G:\text{subgrid}}\bigwedge_{(i,j)\neq(i',j')}\bigwedge_{k=1}^{9}\neg s_{ijk} \vee \neg s_{i'j'k}$$

Uniqueness of Solution

QUESTION

how to check whether there is exactly one solution

Uniqueness of Solution

QUESTION

how to check whether there is exactly one solution

SOLUTION

check whether there is no other solution

Uniqueness of Solution

QUESTION

how to check whether there is exactly one solution

SOLUTION

check whether there is no other solution

THEOREM

let ϕ be CNF encoding, and α its satisfiable assignment

solution is unique if and only if

$$\phi \wedge \left(\Big(\bigvee_{\alpha(x)=T} \neg x \Big) \vee \Big(\bigvee_{\alpha(x)=F} x \Big) \right)$$

is satisfiable

Encoding Techniques

SAT Encoding

- modern SAT solvers are extremely fast

SAT Encoding

- modern SAT solvers are extremely fast
- how to translate problem to CNF?

Quiz

find equivalent CNFs

- $(x \wedge y) \ \vee \ \neg(u \wedge v)$

Quiz

find equivalent CNFs

- $(x \wedge y) \ \vee \ \neg(u \wedge v)$
- $\bigvee_{i=1}^{n} (x_i \wedge y_i)$

Quiz

find equivalent CNFs

- $(x \wedge y) \ \vee \ \neg(u \wedge v)$
- $\bigvee_{i=1}^{n}(x_i \wedge y_i)$

how to avoid exponential blow up? ☞ Tseitin conversion

Tseitin's transformation

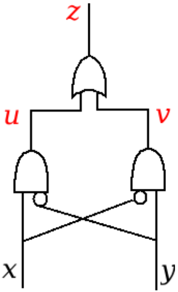
$x \leftrightarrow (y \wedge z) = (\neg x \vee y) \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z \vee x)$ CNF

$x \leftrightarrow (y \vee z) = (x \vee \neg y) \wedge (x \vee \neg z) \wedge (y \vee z \vee \neg x)$ CNF

Tseitin's transformation

$$x \leftrightarrow (y \wedge z) = (\neg x \vee y) \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z \vee x) \qquad \text{CNF}$$

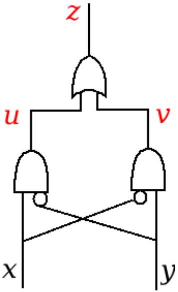
$$x \leftrightarrow (y \vee z) = (x \vee \neg y) \wedge (x \vee \neg z) \wedge (y \vee z \vee \neg x) \qquad \text{CNF}$$



$$\begin{array}{rl} & \models (x \wedge \neg y) \vee (\neg x \wedge y) \\ \Leftrightarrow & \models z \text{ where } \begin{cases} z = u \vee v \\ u = x \wedge \neg y \\ v = \neg x \wedge y \end{cases} \\ \Leftrightarrow & \models z \wedge \bigwedge \left\{ \begin{array}{l} z \leftrightarrow (u \vee v), \\ u \leftrightarrow (x \wedge \neg y), \\ v \leftrightarrow (\neg x \wedge y) \end{array} \right\} \end{array}$$

Tseitin's transformation

$$x \leftrightarrow (y \wedge z) = (\neg x \vee y) \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z \vee x) \qquad \text{CNF}$$
$$x \leftrightarrow (y \vee z) = (x \vee \neg y) \wedge (x \vee \neg z) \wedge (y \vee z \vee \neg x) \qquad \text{CNF}$$



$$
\begin{aligned}
& \models (x \wedge \neg y) \vee (\neg x \wedge y) \\
\Leftrightarrow \quad & \models z \text{ where } \begin{cases} z = u \vee v \\ u = x \wedge \neg y \\ v = \neg x \wedge y \end{cases} \\
\Leftrightarrow \quad & \models z \wedge \bigwedge \left\{ \begin{array}{l} z \leftrightarrow (u \vee v), \\ u \leftrightarrow (x \wedge \neg y), \\ v \leftrightarrow (\neg x \wedge y) \end{array} \right\}
\end{aligned}
$$

NOTE

given formula of size n, converted formula is of size $O(n)$

Arithmetic

$\vec{x}_k = (a_k, \ldots, a_1)$ is binary representation of $x < 2^k$

DEFINITION bit encoding

$$
\begin{aligned}
\vec{x}_k = \vec{y}_k \quad &= \quad \bigwedge_{i=1}^{k} (x_k \leftrightarrow y_k) \\
\vec{x}_k > \vec{y}_k \quad &= \quad \begin{cases} x_1 \wedge y_1 & \text{if } k = 1 \\ (x_k \wedge \neg y_k) \vee \big((x_k \leftrightarrow y_k) \wedge \vec{x}_{k-1} > \vec{y}_{k-1}\big) & \text{if } k > 1 \end{cases} \\
\vec{x}_k + \vec{y}_k \quad &= \quad (c_k, s_k, \ldots, s_1)
\end{aligned}
$$

where

$$
c_0 = \bot \qquad
\begin{aligned}
c_i &= (x_i \wedge y_i) \vee (x_i \wedge c_{i-1}) \vee (y_i \wedge c_{i-1}) && \text{for } i \geqslant 1 \\
s_i &= x_i \oplus y_i \oplus c_{i-1} && \text{for } i \geqslant 1
\end{aligned}
$$

More...

many encodable mathematical objects:

- finite sets

More...

many encodable mathematical objects:

- finite sets
- order $>$ on finite set

More...

many encodable mathematical objects:

- finite sets
- order $>$ on finite set
- graphs

More...

many encodable mathematical objects:

- finite sets
- order $>$ on finite set
- graphs
- ...

DPLL Algorithm

Implementation

many SAT solvers use DPLL algorithm

Implementation

many SAT solvers use DPLL algorithm

- decision, BCP, conflict analysis, clause learning

Implementation

many SAT solvers use DPLL algorithm

- decision, BCP, conflict analysis, clause learning
- two-watched literal (for BCP)

Boolean Constraint Propagation

current assignment:

$$x \mapsto T, y \mapsto F$$

current decision: $z \mapsto T$

$$\neg x \vee z \vee b \vee c$$

$$\neg x \vee y \vee \neg z \vee \neg w$$

$$w \vee y \vee a$$

$$\neg a \vee z \vee b \vee y$$

$$\neg x \vee c \vee d \vee e$$

Boolean Constraint Propagation

current assignment:

$$x \mapsto T, y \mapsto F$$

current decision: $z \mapsto T$

$\neg x \vee z \vee b \vee c$ ✓

$\neg x \vee y \vee \neg z \vee \neg w$

$w \vee y \vee a$

$\neg a \vee z \vee b \vee y$

$\neg x \vee c \vee d \vee e$

Boolean Constraint Propagation

current assignment:

$$x \mapsto T, y \mapsto F$$

current decision: $z \mapsto T$

$\neg x \vee z \vee b \vee c$	✓
$\neg x \vee y \vee \neg z \vee \neg w$	$w \mapsto F$
$w \vee y \vee a$	
$\neg a \vee z \vee b \vee y$	
$\neg x \vee c \vee d \vee e$	

Boolean Constraint Propagation

current assignment:

$$x \mapsto T, y \mapsto F$$

current decision: $z \mapsto T$

$\neg x \vee z \vee b \vee c$	✓
$\neg x \vee y \vee \neg z \vee \neg w$	$w \mapsto F$
$w \vee y \vee a$	–
$\neg a \vee z \vee b \vee y$	–
$\neg x \vee c \vee d \vee e$	–

Two Watch Literals

current assignment:

$$x \mapsto T, y \mapsto F$$

current decision: $z \mapsto T$

$$\neg x \vee z \vee \boxed{b} \vee \boxed{c}$$

$$\neg x \vee y \vee \boxed{\neg z} \vee \boxed{\neg w}$$

$$\boxed{w} \vee \boxed{y} \vee a$$

$$\boxed{\neg a} \vee z \vee \boxed{b} \vee y$$

$$\neg x \vee \boxed{c} \vee \boxed{d} \vee e$$

Conflict-Directed Backtracking

current decision: $x_1 \mapsto T@6$ (T is assigned to x_1 at 6th decision)

$$
\begin{tikzcd}
 & & x_{10} \mapsto F@3 \arrow[dr] & & \\
 & x_2 \mapsto T@6 \arrow[dr] & & x_5 \mapsto T@6 \arrow[dr] & \\
x_1 \mapsto T@6 \arrow[ur] \arrow[dr] & & x_4 \mapsto T@6 \arrow[ur] \arrow[dr] & & \bot \\
 & x_3 \mapsto T@6 \arrow[ur] & & x_5 \mapsto F@6 \arrow[ur] & \\
x_9 \mapsto F@1 \arrow[ur] & & x_{11} \mapsto F@3 \arrow[ur] & &
\end{tikzcd}
$$

- if $x_1 \neq \bot$, back to level 3
- learned clause: $\neg x_1 \vee x_9 \vee x_{10} \vee x_{11}$

Implementation II

many SAT solvers use DPLL algorithm

Implementation II

many SAT solvers use DPLL algorithm

- restart rather than backtrack

Implementation II

many SAT solvers use DPLL algorithm

- restart rather than backtrack
- quick restart with e.g. 32 decisions

Implementation II

many SAT solvers use DPLL algorithm

- restart rather than backtrack
- quick restart with e.g. 32 decisions
- glue clauses (generalisation of unit clauses)

Termination

Term Rewriting

- pair of terms $\ell \to r$ is rewrite rule if ℓ is non-variable and $\mathsf{Var}(r) \subseteq \mathsf{Var}(\ell)$

Term Rewriting

- pair of terms $\ell \to r$ is rewrite rule if ℓ is non-variable and $\mathsf{Var}(r) \subseteq \mathsf{Var}(\ell)$
- term rewrite system (TRS) is set of rewrite rules

Term Rewriting

- pair of terms $\ell \to r$ is rewrite rule if ℓ is non-variable and $\mathsf{Var}(r) \subseteq \mathsf{Var}(\ell)$
- term rewrite system (TRS) is set of rewrite rules
- (rewrite relation) $s \to_{\mathcal{R}} t$ if $\exists \ell \to r \in \mathcal{R},$ context $C,$ substitution σ : $s = C[\ell\sigma] \ \wedge \ t = C[r\sigma]$

Term Rewriting

- pair of terms $\ell \to r$ is rewrite rule if ℓ is non-variable and $\mathsf{Var}(r) \subseteq \mathsf{Var}(\ell)$
- term rewrite system (TRS) is set of rewrite rules
- (rewrite relation) $s \to_{\mathcal{R}} t$ if $\exists \ell \to r \in \mathcal{R}$, context C, substitution σ : $s = C[\ell\sigma] \ \wedge \ t = C[r\sigma]$

Example

TRS $\mathcal{R}$

$$x + 0 \to x \qquad x + \mathsf{s}(y) \to \mathsf{s}(x + y)$$
$$x \times 0 \to 0 \qquad x \times \mathsf{s}(y) \to x \times y + x$$

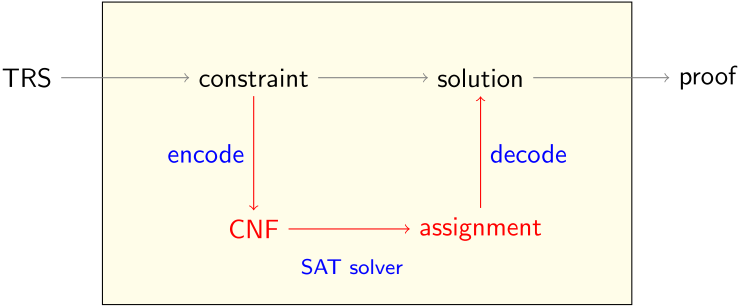
rewriting

$$\begin{aligned} \mathsf{s}(0) \times \mathsf{s}(0) &\to_{\mathcal{R}} \mathsf{s}(0) \times 0 + \mathsf{s}(0) \\ &\to_{\mathcal{R}} 0 + \mathsf{s}(0) \\ &\to_{\mathcal{R}} \mathsf{s}(0 + 0) \\ &\to_{\mathcal{R}} \mathsf{s}(0) \qquad \text{normal form} \end{aligned}$$

APPLICATIONS

- verification for functional programming
- theorem proving
- code optimization in compilers
- symbolic computation in mathematics
- ...

Implementation of Termination Tools



Precedence Termination

precedence $>$ is strict order on function symbols

Precedence Termination

precedence $>$ is strict order on function symbols

Definition

$\ell >_{\mathsf{prec}} r$ if $\mathsf{Var}(\ell) \supseteq \mathsf{Var}(r)$, $\ell = f(\ldots)$, and $f > g$ for all functions g in r

Precedence Termination

precedence $>$ is strict order on function symbols

Definition

$\ell >_{\mathsf{prec}} r$ if $\mathsf{Var}(\ell) \supseteq \mathsf{Var}(r)$, $\ell = f(\ldots)$, and $f > g$ for all functions g in r

Theorem

finite TRS $\mathcal{R}$ is terminating if $\mathcal{R} \subseteq >_{\mathsf{prec}}$ for some precedence $>$

Encoding of Precedence Termination Problem

Encoding of Precedence Termination Problem

Corollary

assume $f > g$ stands for propositional variable.

finite TRS $\mathcal{R}$ is terminating if $\vDash O \wedge I \wedge T$, where

$$O = \bigwedge_{\ell \to r \in \mathcal{R}} (\ell >_{\mathsf{prec}} r)$$

$$I = \bigwedge_{f \in \mathcal{F}} \neg(f > f)$$

$$T = \bigwedge_{f,g,h \in \mathcal{F}} ((f > g) \wedge (g > h) \to (f > h))$$

Encoding of Precedence Termination Problem

Corollary

assume $f > g$ stands for propositional variable.

finite TRS $\mathcal{R}$ is terminating if $\vDash O \wedge I \wedge T$, where

$$O = \bigwedge_{\ell \to r \in \mathcal{R}} (\ell >_{\mathsf{prec}} r)$$

$$I = \bigwedge_{f \in \mathcal{F}} \neg(f > f)$$

$$T = \bigwedge_{f,g,h \in \mathcal{F}} ((f > g) \wedge (g > h) \to (f > h))$$

NOTE

size of T is $O(n^3)$, where $n = |\mathcal{F}|$ (number of function symbols)

Example

prove termination of TRS

$$\begin{aligned}
\mathsf{not}(x) &\to \mathsf{if}(x, \mathsf{false}, \mathsf{true}) \\
\mathsf{and}(x, y) &\to \mathsf{if}(x, y, \mathsf{false}) \\
\mathsf{or}(x, y) &\to \mathsf{if}(x, \mathsf{true}, y) \\
\mathsf{equiv}(x, y) &\to \mathsf{if}(x, y, \mathsf{not}(y))
\end{aligned}
\qquad
\begin{aligned}
\mathsf{if}(\mathsf{true}, x, y) &\to x \\
\mathsf{if}(\mathsf{false}, x, y) &\to y
\end{aligned}$$

$$
O = \bigwedge \left\{ \begin{array}{rlrl}
\mathsf{not}(x) & >_{\mathsf{prec}} \mathsf{if}(x, \mathsf{false}, \mathsf{true}), & \mathsf{if}(\mathsf{true}, x, y) & >_{\mathsf{prec}} x, \\
\mathsf{and}(x, y) & >_{\mathsf{prec}} \mathsf{if}(x, y, \mathsf{false}), & \mathsf{if}(\mathsf{false}, x, y) & >_{\mathsf{prec}} y, \\
\mathsf{or}(x, y) & >_{\mathsf{prec}} \mathsf{if}(x, \mathsf{true}, y), & & \\
\mathsf{equiv}(x, y) & >_{\mathsf{prec}} \mathsf{if}(x, y, \mathsf{not}(y)) & &
\end{array} \right\}
$$

$$
= \left(\begin{array}{rcrcrc}
\mathsf{not} > \mathsf{if} & \wedge & \mathsf{not} > \mathsf{false} & \wedge & \mathsf{not} > \mathsf{true} & \wedge \\
\mathsf{and} > \mathsf{if} & \wedge & \mathsf{and} > \mathsf{false} & \wedge & & \\
\mathsf{or} > \mathsf{if} & \wedge & \mathsf{or} > \mathsf{true} & \wedge & & \\
\mathsf{equiv} > \mathsf{if} & \wedge & \mathsf{equiv} > \mathsf{not} & & &
\end{array} \right)
$$

$$O = \bigwedge \left\{ \begin{array}{rlrl} \mathsf{not}(x) & >_{\mathsf{prec}} \mathsf{if}(x, \mathsf{false}, \mathsf{true}), & \mathsf{if}(\mathsf{true}, x, y) & >_{\mathsf{prec}} x, \\ \mathsf{and}(x, y) & >_{\mathsf{prec}} \mathsf{if}(x, y, \mathsf{false}), & \mathsf{if}(\mathsf{false}, x, y) & >_{\mathsf{prec}} y, \\ \mathsf{or}(x, y) & >_{\mathsf{prec}} \mathsf{if}(x, \mathsf{true}, y), & & \\ \mathsf{equiv}(x, y) & >_{\mathsf{prec}} \mathsf{if}(x, y, \mathsf{not}(y)) & & \end{array} \right\}$$

$$= \left(\begin{array}{rcrcrc} \mathsf{not} > \mathsf{if} & \wedge & \mathsf{not} > \mathsf{false} & \wedge & \mathsf{not} > \mathsf{true} & \wedge \\ \mathsf{and} > \mathsf{if} & \wedge & \mathsf{and} > \mathsf{false} & \wedge & & \\ \mathsf{or} > \mathsf{if} & \wedge & \mathsf{or} > \mathsf{true} & \wedge & & \\ \mathsf{equiv} > \mathsf{if} & \wedge & \mathsf{equiv} > \mathsf{not} & & & \end{array} \right)$$

$O \wedge I \wedge T$ is satisfiable:

$$\mathsf{and} > \mathsf{or} > \mathsf{equiv} > \mathsf{not} > \mathsf{false} > \mathsf{true} > \mathsf{if}$$

hence TRS is terminating

Example

prove termination of TRS $\mathcal{R}$

$$
\begin{aligned}
x + \mathsf{0} &\to x & \qquad x + \mathsf{s}(y) &\to \mathsf{s}(x + y) \\
x \times \mathsf{0} &\to \mathsf{0} & \qquad x \times \mathsf{s}(y) &\to x \times y + x
\end{aligned}
$$

Example

prove termination of TRS $\mathcal{R}$

$$x + \mathsf{0} \rightarrow x \qquad x + \mathsf{s}(y) \rightarrow \mathsf{s}(x + y)$$
$$x \times \mathsf{0} \rightarrow \mathsf{0} \qquad x \times \mathsf{s}(y) \rightarrow x \times y + x$$

precedence termination does not hold:

$$\begin{aligned} & \mathcal{R} \subseteq >_{\mathsf{prec}} \\ \Leftrightarrow\ & \mathsf{s}(x) + y >_{\mathsf{prec}} \mathsf{s}(x + y) \ \wedge\ \cdots \\ \Leftrightarrow\ & + > \mathsf{s}\ \wedge\ + > +\ \wedge\ \cdots \end{aligned}$$

is unsatisfiable

Lexicographic Path Order

Definition given precedence $>$

$s >_{\mathsf{lpo}} t$ if $s = f(s_1, \ldots, s_m)$, and either $t \in \mathsf{Var}(s)$ or $t = g(t_1, \ldots, t_n)$ and

- $s_i >_{\mathsf{lpo}} t$ or $s_i = t$ for all $1 \leqslant i \leqslant m$,
- $f > g$ and $s >_{\mathsf{lpo}} t_i$ for all $1 \leqslant i \leqslant n$, or
- $f = g$ and there is $1 \leqslant i \leqslant n$ with

  $s_1 = t_1, \ldots, s_{i-1} = t_{i-1}$, $s_i >_{\mathsf{lpo}} t_i$, and $s >_{\mathsf{lpo}} t_{i+1}, \ldots s >_{\mathsf{lpo}} t_n$

Lexicographic Path Order

Definition given precedence $>$

$s >_{\mathsf{lpo}} t$ if $s = f(s_1, \dots, s_m)$, and either $t \in \mathsf{Var}(s)$ or $t = g(t_1, \dots, t_n)$ and

- $s_i >_{\mathsf{lpo}} t$ or $s_i = t$ for all $1 \leqslant i \leqslant m$,
- $f > g$ and $s >_{\mathsf{lpo}} t_i$ for all $1 \leqslant i \leqslant n$, or
- $f = g$ and there is $1 \leqslant i \leqslant n$ with

  $s_1 = t_1, \dots, s_{i-1} = t_{i-1}$, $s_i >_{\mathsf{lpo}} t_i$, and $s >_{\mathsf{lpo}} t_{i+1}, \dots s >_{\mathsf{lpo}} t_n$

Theorem Kamin and Levy, 1980

finite TRS $\mathcal{R}$ is terminating if $\mathcal{R} \subseteq >_{\mathsf{lpo}}$ for some precedence $>$

Lexicographic Path Order

DEFINITION given precedence $>$

$s >_{\mathsf{lpo}} t$ if $s = f(s_1, \ldots, s_m)$, and either $t \in \mathsf{Var}(s)$ or $t = g(t_1, \ldots, t_n)$ and

- $s_i >_{\mathsf{lpo}} t$ or $s_i = t$ for all $1 \leqslant i \leqslant m$,
- $f > g$ and $s >_{\mathsf{lpo}} t_i$ for all $1 \leqslant i \leqslant n$, or
- $f = g$ and there is $1 \leqslant i \leqslant n$ with

  $s_1 = t_1, \ldots, s_{i-1} = t_{i-1}$, $s_i >_{\mathsf{lpo}} t_i$, and $s >_{\mathsf{lpo}} t_{i+1}, \ldots s >_{\mathsf{lpo}} t_n$

THEOREM Kamin and Levy, 1980

finite TRS $\mathcal{R}$ is terminating if $\mathcal{R} \subseteq >_{\mathsf{lpo}}$ for some precedence $>$

SAT encoding is similar to precedence termination

Example

TRS $\mathcal{R}$

$$
\begin{aligned}
x + \mathsf{0} &\to x & \qquad x + \mathsf{s}(y) &\to \mathsf{s}(x + y) \\
x \times \mathsf{0} &\to \mathsf{0} & \qquad x \times \mathsf{s}(y) &\to x \times y + x
\end{aligned}
$$

Example

TRS $\mathcal{R}$

$$
\begin{aligned}
x + \mathsf{0} &\to x & \qquad x + \mathsf{s}(y) &\to \mathsf{s}(x + y) \\
x \times \mathsf{0} &\to \mathsf{0} & \qquad x \times \mathsf{s}(y) &\to x \times y + x
\end{aligned}
$$

SAT solver finds precedence that fulfils $\mathcal{R} \subseteq >_{\mathsf{lpo}}$:

$$\times > + > \mathsf{s} > \mathsf{0}$$

hence $\mathcal{R}$ is terminating

Optimization

Annov, Codish, and Stuckey, RTA 2006

Optimization

Annov, Codish, and Stuckey, RTA 2006

FACT

transitivity constraint is $O(n^3)$

Optimization

Annov, Codish, and Stuckey, RTA 2006

FACT

transitivity constraint is $O(n^3)$

LEMMA

two statements are equivalent

- $\mathcal{R} \subseteq >_{\mathsf{lpo}}$ for some precedence $>$

Optimization

Annov, Codish, and Stuckey, RTA 2006

FACT

transitivity constraint is $O(n^3)$

Lemma

two statements are equivalent

- $\mathcal{R} \subseteq >_{\mathsf{lpo}}$ for some precedence $>$
- $\mathcal{R} \subseteq >_{\mathsf{lpo}}$ for some total precedence $>$

Optimization

Annov, Codish, and Stuckey, RTA 2006

FACT

transitivity constraint is $O(n^3)$

LEMMA

two statements are equivalent

- $\mathcal{R} \subseteq >_{\mathsf{lpo}}$ for some precedence $>$
- $\mathcal{R} \subseteq >_{\mathsf{lpo}}$ for some total precedence $>$

IDEA

total precedence $>$ can be represented by weight assignment

- each function f is variable over $\{0, \ldots, n\}$; use bit-encoding

Optimization

Annov, Codish, and Stuckey, RTA 2006

FACT

transitivity constraint is $O(n^3)$

LEMMA

two statements are equivalent

- $\mathcal{R} \subseteq >_{\mathsf{lpo}}$ for some precedence $>$
- $\mathcal{R} \subseteq >_{\mathsf{lpo}}$ for some total precedence $>$

IDEA

total precedence $>$ can be represented by weight assignment

- each function f is variable over $\{0, \ldots, n\}$; use bit-encoding
- size of constraint is $O(N \log n)$, where N is size of TRS

Knuth-Bendix Orders

DEFINITION

- weight assignment $(w_0, w_f, w_g, \ldots)$ is tuple of real numbers where $f, g, \ldots \in \mathcal{F}$

Knuth-Bendix Orders

DEFINITION

- weight assignment $(w_0, w_f, w_g, \ldots)$ is tuple of real numbers where $f, g, \ldots \in \mathcal{F}$
- weight of term t is

$$w(t) = \begin{cases} w_0 & \text{if } t \text{ is variable} \\ w_f + w(t_1) + \cdots + w(t_n) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

Knuth-Bendix Orders

DEFINITION

- weight assignment $(w_0, w_f, w_g, \ldots)$ is tuple of real numbers where $f, g, \ldots \in \mathcal{F}$
- weight of term t is

$$w(t) = \begin{cases} w_0 & \text{if } t \text{ is variable} \\ w_f + w(t_1) + \cdots + w(t_n) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

- weight assignment is admissible for precedence $>$ if

$$w_f > 0 \qquad \text{or} \qquad f \geqslant g$$

  for all unary functions f and all functions g

Definition Knuth-Bendix order; given weight w and precedence $>$

$s >_{\mathsf{kbo}} t$ if $|s|_x \geqslant |t|_x$ for all variables x and either

DEFINITION Knuth-Bendix order; given weight w and precedence $>$

$s >_{\mathsf{kbo}} t$ if $|s|_x \geqslant |t|_x$ for all variables x and either

- $w(s) > w(t)$, or

Definition Knuth-Bendix order; given weight w and precedence $>$

$s >_{\mathsf{kbo}} t$ if $|s|_x \geqslant |t|_x$ for all variables x and either

- $w(s) > w(t)$, or
- $w(s) = w(t)$ and

Definition Knuth-Bendix order; given weight w and precedence $>$

$s >_{\mathsf{kbo}} t$ if $|s|_x \geqslant |t|_x$ for all variables x and either

- $w(s) > w(t)$, or
- $w(s) = w(t)$ and
  - $s = f^n(t)$ and t is variable for some unary f and $n \geqslant 1$; or

DEFINITION Knuth-Bendix order; given weight w and precedence $>$

$s >_{\mathsf{kbo}} t$ if $|s|_x \geqslant |t|_x$ for all variables x and either

- $w(s) > w(t)$, or
- $w(s) = w(t)$ and
  - $s = f^n(t)$ and t is variable for some unary f and $n \geqslant 1$; or
  - $s = f(s_1, \ldots, s_{i-1}, s_i, \ldots, s_n)$, $t = f(s_1, \ldots, s_{i-1}, t_i, \ldots, t_n)$, and $s_i >_{\mathsf{kbo}} t_i$; or

Definition Knuth-Bendix order; given weight w and precedence $>$

$s >_{\mathsf{kbo}} t$ if $|s|_x \geqslant |t|_x$ for all variables x and either

- $w(s) > w(t)$, or
- $w(s) = w(t)$ and
  - $s = f^n(t)$ and t is variable for some unary f and $n \geqslant 1$; or
  - $s = f(s_1, \ldots, s_{i-1}, s_i, \ldots, s_n)$, $t = f(s_1, \ldots, s_{i-1}, t_i, \ldots, t_n)$, and $s_i >_{\mathsf{kbo}} t_i$; or
  - $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, and $f > g$

Definition Knuth-Bendix order; given weight w and precedence $>$

$s >_{\mathsf{kbo}} t$ if $|s|_x \geqslant |t|_x$ for all variables x and either

- $w(s) > w(t)$, or
- $w(s) = w(t)$ and
  - $s = f^n(t)$ and t is variable for some unary f and $n \geqslant 1$; or
  - $s = f(s_1, \ldots, s_{i-1}, s_i, \ldots, s_n)$, $t = f(s_1, \ldots, s_{i-1}, t_i, \ldots, t_n)$, and $s_i >_{\mathsf{kbo}} t_i$; or
  - $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, and $f > g$

Theorem Knuth and Bendix, 1970; Dershowitz, 1979

finite TRS $\mathcal{R}$ is terminating if $\mathcal{R} \subseteq >_{\mathsf{kbo}}$ for some weight $(w_0, w_f, w_g, \ldots)$ over $\mathbb{R}$ and precedence $>$

THEOREM Hirokawa, Zankl, Middeldorp, JAR 2010

next statements are equivalent

- $\mathcal{R} \subseteq >_{\mathsf{kbo}}$ for some weight w over $\mathbb{R}$ and precedence $>$
- $\mathcal{R} \subseteq >_{\mathsf{kbo}}$ for some weight w over $\{0, 1, \ldots, 2^{2^N}\}$ and precedence $>$

here N is size of $\mathcal{R}$

Theorem Hirokawa, Zankl, Middeldorp, JAR 2010

next statements are equivalent

- $\mathcal{R} \subseteq >_{\mathsf{kbo}}$ for some weight w over $\mathbb{R}$ and precedence $>$
- $\mathcal{R} \subseteq >_{\mathsf{kbo}}$ for some weight w over $\{0, 1, \ldots, 2^{2^N}\}$ and precedence $>$

here N is size of $\mathcal{R}$

NOTE

large weights (> 15) are hardly required

Termination Tools

- termination tools (AProVE, Matchbox, μ-Term, TTT2, VMTL, ...) use SAT/SMT solvers

Termination Tools

- termination tools (AProVE, Matchbox, μ-Term, TTT2, VMTL, ...) use SAT/SMT solvers

- to increase power, termination tools employ transformations

  dependency pair method by Arts and Giesl, TCS 2000

Termination Tools

- termination tools (AProVE, Matchbox, μ-Term, TTT2, VMTL, ...) use SAT/SMT solvers

- to increase power, termination tools employ transformations

  dependency pair method by Arts and Giesl, TCS 2000

- complexity analysers (AProVE, CaT, TCT, ...) use same way

  POP* by Avanzini and Moser (2008)

Summary

- SAT solver and basic encoding techniques

Summary

- SAT solver and basic encoding techniques
- termination analysis

Summary

- SAT solver and basic encoding techniques
- termination analysis
- often SAT encoding approach outperforms dedicated algorithm

Summary

- SAT solver and basic encoding techniques
- termination analysis
- often SAT encoding approach outperforms dedicated algorithm

thank you for your kind attention!