SAT/SMT Solving and Applications in Rewriting

René Thiemann [1] Sarah Winkler [2]

[1]University of Innsbruck

[2]Free University of Bolzano

Outline

1. Solution of Exercise of Session 2
2. Lazy SMT Approach: Overview
3. Application: Polynomial Interpretations
4. Non-Linear (Bit-Vector) Arithmetic
5. Certification
6. Further Reading

Exercise

Develop a LIA encoding that searches for an argument filter π in combination with KBO parameters

$$\ulcorner \pi(s) \succ \pi(t) \urcorner$$

definitions

- $\pi(x) = x$
- $\pi(f(t_1, \ldots, t_n)) = \begin{cases} \pi(t_i), & \text{if } \pi(f) = i \\ f([\pi(t_i) \mid i \leftarrow [1..n], i \in \pi(f)]), & \text{if } \pi(f) \text{ is a set} \end{cases}$
- $w(x) = w_0$
- $w(f(t_1, \ldots, t_n)) = w(f) + w(t_1) + \cdots + w(t_n)$
- $s \succ t$ if $\mathcal{V}(s) \supseteq \mathcal{V}(t) \wedge (w(s) > w(t) \vee w(s) \geq w(t) \wedge$ ... some cases ...)

Solution of Encoding KBO + AF (1/2)

- we use propositional variables $set(f), i \in \pi(f)$ to represent AFs as for LPO
- we use the same constraints to enforce that the AF is well-formed
- if-then-else is written as $\ulcorner if(b,t,e) \urcorner$; it is a short-cut for
  - creating a fresh integer variable i
  - returning i as the result of $\ulcorner if(b,t,e) \urcorner$
  - adding $b \rightarrow i = t$ and $\neg b \rightarrow i = e$ to global constraints
- encode frequency of variable x in term t as integer variable $\ulcorner \#_x(\pi(t)) \urcorner$; add constr.
  - $\ulcorner \#_x(\pi(x)) \urcorner = 1$ and $\ulcorner \#_x(\pi(y)) \urcorner = 0$ if $x \neq y$
  - $\ulcorner \#_x(\pi(f(t_1,\dots,t_n))) \urcorner = \ulcorner if(1 \in \pi(f), \ulcorner \#_x(\pi(t_1)) \urcorner, 0) \urcorner + \dots + \ulcorner if(n \in \pi(f), \ulcorner \#_x(\pi(t_n)) \urcorner, 0) \urcorner$
- now $\mathcal{V}(\pi(s)) \supseteq \mathcal{V}(\pi(t))$ is encoded as $\bigwedge_{x \in \mathcal{V}(t)} \ulcorner \#_x(\pi(s)) \urcorner \geq \ulcorner \#_x(\pi(t)) \urcorner$
- the weight computation is similar using integer variables $\ulcorner w(\pi(t)) \urcorner$
  - $\ulcorner w(\pi(x)) \urcorner = w_0$
  - $\neg set(f) \rightarrow i \in \pi(f) \rightarrow \ulcorner w(\pi(f(t_1,\dots,t_n))) \urcorner = \ulcorner w(\pi(t_i)) \urcorner$
  - $set(f) \rightarrow \ulcorner w(\pi(f(t_1,\dots,t_n))) \urcorner = w(f) +$
    $\ulcorner if(1 \in \pi(f), \ulcorner w(\pi(t_1)) \urcorner, 0) \urcorner + \dots + \ulcorner if(n \in \pi(f), \ulcorner w(\pi(t_n)) \urcorner, 0) \urcorner$

Solution of Encoding KBO + AF (2/2)

- having integer variables $\ulcorner w(\pi(t)) \urcorner$ and an encoding of $\mathcal{V}(\pi(s)) \supseteq \mathcal{V}(\pi(t))$, encoding term comparisons in KBO + AF is now similar to the term comparison of LPO + AF
- additional challenge: admissibility
  - we need to encode $\ulcorner \mathit{unary}(f) \urcorner := \mathit{set}(f) \wedge \ulcorner \mathit{exactlyOne}(1 \in \pi(f), \ldots, n \in \pi(f)) \urcorner$
  - being largest in precedence can be restricted to those symbols g that remain

$$\ulcorner \mathit{unary}(f) \urcorner \to w(f) = 0 \to \bigwedge_{g \neq f} (\mathit{set}(g) \to p(f) > p(g))$$

  - weights for constants need to be adjusted: $\mathit{set}(f) \to (\bigwedge_i \neg(i \in \pi(f))) \to w(f) \geq w_0$
  - no weight restrictions for $w(f)$ apply, whenever $\neg \mathit{set}(f)$

Outline

1. Solution of Exercise of Session 2
2. Lazy SMT Approach: Overview
3. Application: Polynomial Interpretations
4. Non-Linear (Bit-Vector) Arithmetic
5. Certification
6. Further Reading

SMT Solving at a Glance

- DPLL(T) is common approach for SMT solving
- requirement: theory solver for T
  - given conjunction of literals, decide T-satisfiability
- overview of theory solvers
  - LRA: simplex algorithm (Dutertre and de Moura)
    - incremental interface
    - delivers minimal unsatisfiable cores
  - LIA: LRA + branch-and-bound algorithm
    - call simplex on constraints φ
    - if φ is unsat in $\mathbb{Q}$ then return “unsat”
    - if solution delivers $\alpha(x) = q \notin \mathbb{Z}$, then branch on $\varphi \wedge x \leq \lfloor q \rfloor$ “or” $\varphi \wedge x \geq \lceil q \rceil$
    - otherwise, return integer solution
  - many extensions for LIA
  - EUF: congruence closure algorithm
  - combination of theories: Nelson–Oppen, deterministic or nondeterministic version
- due to limited time: omit further details in this course

Outline

1. Solution of Exercise of Session 2
2. Lazy SMT Approach: Overview
3. Application: Polynomial Interpretations
4. Non-Linear (Bit-Vector) Arithmetic
5. Certification
6. Further Reading

Definition (Linear Polynomial Interpretation)

- fix some signature $\mathcal{F}$; choose for each n-ary $f \in \mathcal{F}$ a linear polynomial $p(f)$:

$$p(f) = f_0 + f_1 x_1 + \ldots f_n x_n$$

  such that $f_0 \in \mathbb{N}$ and $f_i \in \mathbb{N} \setminus \{0\}$ for $1 \leq i \leq n$
- interpretation of terms
  - $[\![x]\!] = x$
  - $[\![f(t_1, \ldots, t_n)]\!] = p(f)\{x_1/[\![t_1]\!], \ldots, x_n/[\![t_n]\!]\}$
- definition of order: $s \succ t$ iff $\forall \vec{x}.\ [\![s]\!] > [\![t]\!]$ where variables $\vec{x}$ range over $\mathbb{N}$

Example (Termination of $\{plus(s(x), y) \to s(plus(x, y)); \quad plus(0, y) \to y\}$)

- choose $p(0) = 5$ and $p(\mathsf{plus}) = 2 \cdot x_1 + x_2$ and $p(\mathsf{s}) = 1 + x_1$
- first rule: $2 \cdot (1 + x) + y > 1 + 2 \cdot x + y$
- second rule: $2 \cdot 5 + y > y$

Definition (Encoding for Linear Polynomial Interpretations)

- fix some signature $\mathcal{F}$; encode for each n-ary $f \in \mathcal{F}$ a linear polynomial $p(f)$ using (SMT) integer variables f_i:

$$p(f) = f_0 + f_1 x_1 + \dots f_n x_n$$

  and add constraints $f_0 \geq 0$ and $f_i \geq 1$ for $1 \leq i \leq n$
- compute $[\![t]\!]$ symbolically and then compare coefficients for each variable:

$$a + bx + cy + \dots > a' + b'x + c'y + \dots \equiv \underbrace{a > a' \land b \geq b' \land c \geq c' \land \dots}_{\text{SMT constraint}}$$

Example (Constraint of first rule $plus(s(x), y) \to s(plus(x, y))$)

$$\begin{aligned}
& \mathsf{plus}_0 + \mathsf{plus}_1(\mathsf{s}_0 + \mathsf{s}_1 x) + \mathsf{plus}_2 y > \mathsf{s}_0 + \mathsf{s}_1(\mathsf{plus}_0 + \mathsf{plus}_1 x + \mathsf{plus}_2 y) \\
\equiv\; & (\mathsf{plus}_0 + \mathsf{plus}_1 \mathsf{s}_0) + \mathsf{plus}_1 \mathsf{s}_1 x + \mathsf{plus}_2 y > (\mathsf{s}_0 + \mathsf{s}_1 \mathsf{plus}_0) + \mathsf{s}_1 \mathsf{plus}_1 x + \mathsf{s}_1 \mathsf{plus}_2 y \\
\equiv\; & \mathsf{plus}_0 + \mathsf{plus}_1 \mathsf{s}_0 > \mathsf{s}_0 + \mathsf{s}_1 \mathsf{plus}_0 \land \mathsf{plus}_1 \mathsf{s}_1 \geq \mathsf{s}_1 \mathsf{plus}_1 \land \mathsf{plus}_2 \geq \mathsf{s}_1 \mathsf{plus}_2 \qquad \textit{SMT constr.}
\end{aligned}$$

Exercise

- design an optimized encoder for polynomial constraints; you should consider a weakly monotone setting where the condition
$$f_0 \geq 0 \text{ and } f_i \geq 1 \text{ for all } 1 \leq i \leq n$$
is weakened to
$$f_i \geq 0 \text{ for } 0 \leq i \leq n$$
- test your encoding on the following term constraints

$$\begin{aligned}
\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) &\gtrsim \mathsf{minus}(x, y) \\
\mathsf{minus}(x, \mathsf{0}) &\gtrsim x \\
\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) &\succ \mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y))
\end{aligned}$$

where $s \gtrsim t$ is defined as $\forall \vec{x}. [\![s]\!] \geq [\![t]\!]$

Outline

1. Solution of Exercise of Session 2
2. Lazy SMT Approach: Overview
3. Application: Polynomial Interpretations
4. Non-Linear (Bit-Vector) Arithmetic
5. Certification
6. Further Reading

A Problem

- resulting constraints are non-linear integer constraints
- problem: NIA is undecidable
- encoding does not matter: linear polynomial termination is undecidable

A Solution

- restrict search space: often small coefficients suffice, e.g., $f_i \in \{0, \ldots, 3\}$, i.e., each f_i is a 2-bit number
- on numbers with fixed bit-width, one can perform bit-vector arithmetic
- basic idea: encode hardware adders, multipliers, comparisons, etc. into SAT
- SMT theory QF_BV: bitvector arithmetic uses eager approach for SMT solving
- result: obtain incomplete NIA solver via decidable BV theory

Handling Overflows

- BV differs from NIA in that overflows may happen
- $3 > 3 + 3$ if everything is evaluated using 2-bit unsigned numbers
- overflows must not happen in order to simulate NIA computations in BV
- two solutions: choose enough bits or forbid overflows

Handling Overflows: Choose Enough Bits

- consider linear polynomial interpretation example
- non-linear formula is known

$$\mathsf{plus}_0 + \mathsf{plus}_1 \mathsf{s}_0 > \mathsf{s}_0 + \mathsf{s}_1 \mathsf{plus}_0 \wedge \mathsf{plus}_2 \geq \mathsf{s}_1 \mathsf{plus}_2$$

- given b bits as input size for variables, we can bound bit-sizes of intermediate expressions

$$\underbrace{\underbrace{\underbrace{\mathsf{plus}_0}_{b} + \underbrace{\underbrace{\mathsf{plus}_1}_{b}\,\underbrace{\mathsf{s}_0}_{b}}_{2b}}_{2b+1} > \underbrace{\underbrace{\mathsf{s}_0}_{b} + \underbrace{\underbrace{\mathsf{s}_1}_{b}\,\underbrace{\mathsf{plus}_0}_{b}}_{2b}}_{2b+1}}_{2b+1} \wedge \underbrace{\underbrace{\mathsf{plus}_2}_{b} \geq \underbrace{\underbrace{\mathsf{s}_1}_{b}\,\underbrace{\mathsf{plus}_2}_{b}}_{2b}}_{2b}$$

- hence, one just has to perform each bit-vector operation with enough bits

Handling Overflows: Choose Enough Bits, Optimized

- computing upper bounds on values results in better bit-bounds

$$\underbrace{\underbrace{\underbrace{\mathsf{plus}_0}_{2^b-1} + \underbrace{\underbrace{\mathsf{plus}_1}_{2^b-1}\,\underbrace{\mathsf{s}_0}_{2^b-1}}_{(2^b-1)^2}}_{(2^b-1)^2+2^b-1} > \underbrace{\underbrace{\mathsf{s}_0}_{2^b-1} + \underbrace{\underbrace{\mathsf{s}_1}_{2^b-1}\,\underbrace{\mathsf{plus}_0}_{2^b-1}}_{(2^b-1)^2}}_{(2^b-1)^2+2^b-1}}_{(2^b-1)^2+2^b-1} \wedge \underbrace{\underbrace{\mathsf{plus}_2}_{2^b-1} \geq \underbrace{\underbrace{\mathsf{s}_1}_{2^b-1}\,\underbrace{\mathsf{plus}_2}_{2^b-1}}_{(2^b-1)^2}}_{(2^b-1)^2}$$

- previous slide: $2b + 1$ bits (7 bits, if $b = 3$)
- this slide: $\lceil \log_2((2^b - 1)^2 + 2^b - 1) \rceil$ bits (6 bits, if $b = 3$)

Handling Overflows: Forbid Overflows

- using always enough bits might be expensive
- alternative
  - select a fixed number of b bits for inputs
  - select a fixed number of c bits for calculations, $b \leq c$
  - all intermediate expressions in formula must be representable with c bits
  - add constraints that ensure that no overflow happens
  - examples
    - perform addition with $c + 1$ bits and demand that highest bit of result is 0
    - perform multiplication with $2c$ bits and demand that the c highest bits of result are all 0
    - encode multiplication using c bits with dedicated overflow bit
    - perform multiplication $x \cdot y$ with c bits and demand
      "position of first 1-bit in x + position of first 1-bit of $y \leq c$"
  - coarse constraint for $c = 3$

$$
\begin{aligned}
&x_3x_2x_1x_0 \cdot y_3y_2y_1y_0 = z_3z_2z_1z_0 \land \\
&(\neg x_3 \land \neg x_2 \land \neg x_1 \quad \lor \quad \neg x_3 \land \neg x_2 \land \neg y_3 \land \neg y_2 \quad \lor \quad \neg y_3 \land \neg y_2 \land \neg y_1)
\end{aligned}
$$

Outline

1. Solution of Exercise of Session 2
2. Lazy SMT Approach: Overview
3. Application: Polynomial Interpretations
4. Non-Linear (Bit-Vector) Arithmetic
5. Certification
6. Further Reading

Current State

- SAT and SMT encodings are useful for proof search
  - often easy to design encoding
  - benefit from powerful SAT and SMT solvers
  - here: focus on termination proving for TRSs
- problem: reliability
  - SAT and SMT solver might be buggy
  - language binding might be buggy
  - encoding might contain some mistake
  - implementation of encoding might be buggy
- solution: certification
  - validate generated proofs

Certification – The Easy Direction

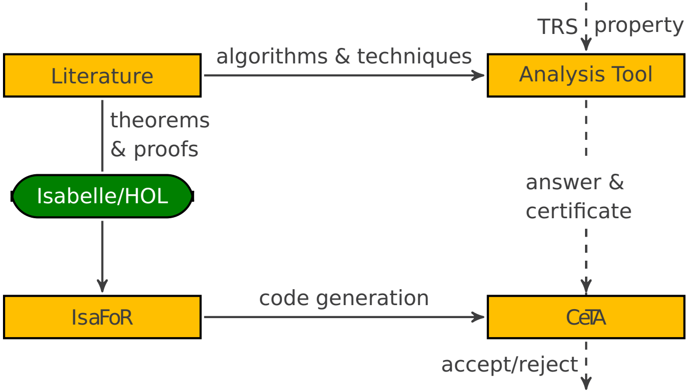
- all examples so far aimed at finding satisfying assignments
  - find parameters of KBO, LPO and polynomial interpretations
  - find argument filters
- every satisfying assignment leads to concrete instance of that term order, e.g.:
  - KBO with $w_0 = 5$, $w(\mathsf{plus}) = 2$, $p(\mathsf{plus}) > p(\mathsf{s})$, ...
  - AF with $\pi(\mathsf{minus}) = 1, \pi(\mathsf{div}) = \{1\}$, ...
- given a concrete term order $\succ$, it is often trivial to check correct application
  - check $\ell \succ r$ for all $\ell \to r \in \mathcal{R}$
  - check admissibility of KBO parameters, ...
- the corresponding algorithms
  - do not require any encodings or any invocation of a SAT or SMT solver
  - are often simple to implement and are therefore less likely to be bugged
- AProVE (in 2007) contained two independent implementations for several orders
  1. an optimized search engine
  2. a simple implementation for concrete instances; used for internal validation

Certification – Trust the Validation Algorithm

- remaining problem
  - what if certification algorithm is buggy?
  - what if definition of order itself is buggy?
- solution: formal verification
  - formal verification: formal proof using proof assistant such as Isabelle, Coq, Lean, ...
  - verify correctness of certification algorithm
  - verify properties of order, e.g., "LPO is reduction order"
- both in termination competition and confluence competition, validity of several proofs is checked by formally verified certifier: CeTA
  - several: not all proofs are supported CeTA
  - CeTA: Certified Tool Assertions, developed in Innsbruck
- example: all CR/COM/INF-tags in ARI-database are validated by CeTA

`https://ari-cops.uibk.ac.at/ARI/?m=results`

Formally Verified Certification



`http://cl-informatik.uibk.ac.at/software/ceta/`

Certification with CeTA

- about CeTA
  - CeTA is just a Haskell program
  - no external libraries required
  - easy to use
    - `ghc --make Main.hs -o ceta`
    - `ceta cpf_proof.xml`
- CPF: Certification Problem Format
  - XML
  - domain-specific proof format, no Isabelle knowledge required
  - covers term rewriting and integer transition systems

- CPF generation is usually straight-forward; in miniTT: 83 lines, cf. `Proof.hs`
- result of `miniTT cpf kbo plus.ari > kbo_plus.xml`


```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="xml/cpf3HTML.xsl"?>
<certificationProblem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="xml/cpf3.xsd"><cpfVersion>3.0</cpfVersion><lookupTables/>
<input><trsInput><trs><rules><rule><funapp><name>plus</name><funapp><name>s</name><var>n
</var></funapp><var>m</var></funapp><funapp><name>s</name><funapp><name>plus</name><var>n
</var><var>m</var></funapp></funapp></rule><rule><funapp><name>plus</name><funapp><name>0
</name></funapp><var>m</var></funapp><var>m</var></rule></rules></trs></trsInput></input>
<property><termination/></property><answer><yes/></answer><proof><trsTerminationProof>
<ruleRemoval><knuthBendixOrder><w0>1</w0><precedenceWeight><precedenceWeightEntry><name>0
</name><arity>0</arity><precedence>0</precedence><weight>1</weight></precedenceWeightEntry
><precedenceWeightEntry><name>plus</name><arity>2</arity><precedence>1</precedence><weight
>0</weight></precedenceWeightEntry><precedenceWeightEntry><name>s</name><arity>1</arity>
<precedence>0</precedence><weight>1</weight></precedenceWeightEntry></precedenceWeight>
</knuthBendixOrder><trs><rules><rule><funapp><name>plus</name><funapp><name>s</name><var>n
</var></funapp><var>m</var></funapp><funapp><name>s</name><funapp><name>plus</name><var>n
</var><var>m</var></funapp></funapp></rule><rule><funapp><name>plus</name><funapp><name>0
</name></funapp><var>m</var></funapp><var>m</var></rule></rules></trs><trsTerminationProof
><rIsEmpty/></trsTerminationProof></ruleRemoval></trsTerminationProof></proof>
</certificationProblem>
```

Adding Indentation

```
<certificationProblem>
... <input><trs> ... <property><termination> ... <answer><yes> ...
      <knuthBendixOrder>
       <w0>1</w0>
       <precedenceWeight>
        <precedenceWeightEntry>
         <name>0</name>
         <arity>0</arity>
         <precedence>0</precedence>
         <weight>1</weight>
        </precedenceWeightEntry>
        <precedenceWeightEntry>
         <name>plus</name>
         <arity>2</arity>
         <precedence>1</precedence>
         <weight>0</weight>
        </precedenceWeightEntry>
        ...
       </precedenceWeight>
      </knuthBendixOrder>
...
</certificationProblem>
```

CPF is Human Readable

- conversion to HTML: `xsltproc cpf3HTML.xsl kbo_plus.xml > kbo_plus.html`

The rewrite relation of the following TRS is considered.

$$\mathsf{plus}(\mathsf{s}(n),m) \to \mathsf{s}(\mathsf{plus}(n,m))$$
$$\mathsf{plus}(\mathsf{O},m) \to m$$

Property / Task

Prove or disprove termination.

Answer / Result

Yes.

Proof (by miniTT)

1 Rule Removal

Using the Knuth Bendix order with w0 = 1 and the following precedence and weight functions

$$\begin{aligned} \mathrm{prec}(\mathsf{plus}) &= 1 & \mathrm{weight}(\mathsf{plus}) &= 0 \\ \mathrm{prec}(\mathsf{s}) &= 0 & \mathrm{weight}(\mathsf{s}) &= 1 \\ \mathrm{prec}(\mathsf{O}) &= 0 & \mathrm{weight}(\mathsf{O}) &= 1 \end{aligned}$$

all of the following rules can be deleted.

$$\mathsf{plus}(\mathsf{s}(n),m) \to \mathsf{s}(\mathsf{plus}(n,m))$$
$$\mathsf{plus}(\mathsf{O},m) \to m$$

Beyond Straight-Forward Certification

- IsaFoR is formalization of soundness of CeTA
- in particular, it contains
  - definitions of KBO, LPO, . . . ,
  - formal proofs that these order have good properties, and
  - verified algorithms for checking certificates
- fact: tools often use optimized versions of orders, e.g.
  - quasi-precedences
  - $x \succsim c$ if c is constant with least precedence
- sometimes these “optimizations” break soundness
  - optimized RPO in AProVE was not closed under substitutions
  - optimized WPO in NaTT was not transitive
  - various incorrect versions of AC-KBO
- many of these problems have been resolved by formal proofs
  - design of IsaFoR: try to include all optimizations to accept many generated proofs
  - example for “optimized RPO”: add further inference rule that restores closure properties

Certification – The Hard Direction

- sometimes a successful proof requires unsatisfiability proofs
- example: termination proofs using weighted path orders (WPO) with max-poly interpretations
  - assign to each n-ary function symbol a max-polynomial, i.e., an arithmetic expression of $\mathcal{T}(\mathbb{N} \cup \{+, \times, \mathsf{max}\}, \{x_1, \ldots, x_n\})$
  - example

$$[\![\text{if-then-else}]\!](x, y, z) = \mathsf{max}(y, z)$$
$$[\![\text{Cons}]\!](x, xs) = 1 + xs$$

- problem: how to check $\forall \vec{x}.\ [\![s]\!] > [\![t]\!]$, i.e., compare max-polynomials?
- solution: show that $\neg([\![s]\!] > [\![t]\!])$ is unsatisfiable

Handling Max-Polynomials in CeTA

- normalize max-polynomials

$$\max(x, y) + z \to \max(x + z, y + z) \qquad \max(x, y) \cdot z \to \max(x \cdot z, y \cdot z)$$

  result has form $\max_{i=1}^{m} p_i$ where each p_i is ordinary polynomial
- transform term-constraint into formula over natural number arithmetic

$$[\![s]\!] > [\![t]\!] \iff \max_{i=1}^{m} p_i > \max_{j=1}^{k} q_j \iff \bigwedge_{j=1}^{k} \bigvee_{i=1}^{m} p_i > q_j$$

- check unsatisfiability of following formula by verified SMT solver for LIA

$$\neg \left(\bigwedge_{x \in Vars(s,t)} x \geq 0 \to \bigwedge_{j=1}^{k} \bigvee_{i=1}^{m} p_i > q_j \right)$$

- own solver avoids bulky certificates: $\mathcal{O}(n^2)$ many $>$-compares for each WPO-constr.

Outline

1. Solution of Exercise of Session 2
2. Lazy SMT Approach: Overview
3. Application: Polynomial Interpretations
4. Non-Linear (Bit-Vector) Arithmetic
5. Certification
6. Further Reading

Further Reading

- Daniel Kroening and Ofer Strichman
  Decision Procedures – An Algorithmic Point of View, Second Edition
  Texts in Theoretical Computer Science, An EATCS Series, Springer, 2016
- Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl
  SAT Solving for Termination Analysis with Polynomial Interpretations
  Proceedings SAT 2007, LNCS 4501, pp. 340 – 354, 2007
- René Thiemann and Christian Sternagel,
  Certification of Termination Proofs Using CeTA
  Proceedings TPHOLs 2009, LNCS 5674, pp. 452 – 468, 2009
- Alexander Lochmann and Christian Sternagel,
  Certified ACKBO
  Proceedings CPP 2019, ACM, pp. 144 – 151, 2019
- René Thiemann, Jonas Schöpf, Christian Sternagel, and Akihisa Yamada,
  Certifying the Weighted Path Order (Invited Talk)
  Proceedings FSCD 2020, LIPIcs 165, pp. 4:1 – 4:20, 2020