Synthesis of Polynomial Interpretations

Petar Maksimović, Radovan Obradović

petarmax@mi.sanu.ac.rs
radovan.obradovic@gmail.com

Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade
Faculty of Technical Sciences, University of Novi Sad

FIT Summer School
Novi Sad
26. 06. 2009.

Overview

Motivation

- Polynomial interpretations – now widely used in the development of tools for proving termination of TRSs
- One cannot rely on user-defined polynomial interpretations for generating polynomial orderings – some form of automatization is required

Motivation

- Polynomial interpretations – now widely used in the development of tools for proving termination of TRSs
- One cannot rely on user-defined polynomial interpretations for generating polynomial orderings – some form of automatization is required

- Main question: How to synthesize appropriate polynomial orderings for a given TRS?

Preliminaries

Definition (Main concepts)

- $\mathcal{T}(\mathcal{F}, X)$ – a set of terms over signature $\mathcal{F}$ and variables X;
- Term ordering – a pair $(\succeq, >)$ of relations over $\mathcal{T}(\mathcal{F}, X)$, such that
  - $\succeq$ is a quasi-ordering: reflexive and transitive;
  - $>$ is a strict ordering: irreflexive and transitive;
  - $\succeq \cdot > \Rightarrow >$ or $> \cdot \succeq \Rightarrow >$;
- A term ordering is *well-founded* if there is no infinite sequence of terms $t_1 > t_2 > \ldots$;
- A term ordering is *stable* if both $\succeq$ and $>$ are stable under substitution;
- For $f \in \mathcal{F}$ of arity $n \geq 1$, a relation $\mathcal{R}$ is monotonic with reference to the i-th argument of f, $1 \leq 1 \leq n$, if $t \mathcal{R} u$ implies $f(v_1, \ldots, v_{i-1}, t, v_{i+1}, \ldots, v_n) \mathcal{R} f(v_1, \ldots, v_{i-1}, u, v_{i+1}, \ldots, v_n)$, for any $t, u, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n \in \mathcal{T}(\mathcal{F}, X)$;

Preliminaries

Definition (Main concepts)

A term ordering is:

- *weakly monotonic* if $\geq$ is monotonic with reference to all arguments of all function symbols;
- *strictly monotonic* if $>$ is monotonic with reference to all arguments of all function symbols;
- called a *weak reduction ordering* if it is well-founded, stable and weakly monotonic;
- called a *strict reduction ordering* if it is well-founded, stable and strictly monotonic;

Preliminaries

Theorem (Manna and Ness termination criterion)

If, for a given TRS R, there exists a strict reduction ordering $(\succeq, >)$, such that for every rewriting rule $l \to r \in R$, it holds that $l > r$, then R is terminating.

Preliminaries

Theorem (Manna and Ness termination criterion)

If, for a given TRS R, there exists a strict reduction ordering $(\succeq, >)$, such that for every rewriting rule $l \to r \in R$, it holds that $l > r$, then R is terminating.

Theorem (Simple dependency pairs criterion)

If, for a given TRS R, there exists a weak reduction ordering $(\succeq, >)$, such that

- *for each rule $l \to r \in R$, $l \succeq r$;*
- *for each dependency pair $\langle u, v \rangle$ of R, $u > v$;*

then R is terminating.

Term orderings on ground terms

- D - an arbitrary non-empty domain;
- $\geq_D$ - an ordering on D;
- $>_D$ - defined as $\geq_D - \leq_D$;

Definition (Term ordering on ground terms)

Let φ be a function mapping each ground term $t \in \mathcal{T}(\mathcal{F}, X)$ to an element of D. The relations $\succeq_\varphi$ and $>_\varphi$ generated by φ are defined by:

$$t_1 \succeq_\varphi t_2 \text{ iff } \varphi(t_1) \geq_D \varphi(t_2)$$
$$t_1 >_\varphi t_2 \text{ iff } \varphi(t_1) >_D \varphi(t_2)$$

Term orderings on ground terms

- D - an arbitrary non-empty domain;
- $\geq_D$ - an ordering on D;
- $>_D$ - defined as $\geq_D - \leq_D$;

Definition (Term ordering on ground terms)

Let φ be a function mapping each ground term $t \in \mathcal{T}(\mathcal{F}, X)$ to an element of D. The relations $\succeq_\varphi$ and $>_\varphi$ generated by φ are defined by:

$$t_1 \succeq_\varphi t_2 \text{ iff } \varphi(t_1) \geq_D \varphi(t_2)$$
$$t_1 >_\varphi t_2 \text{ iff } \varphi(t_1) >_D \varphi(t_2)$$

Lemma (Well-foundedness of $(\succeq_\varphi, >_\varphi)$)

$(\succeq_\varphi, >_\varphi)$ is a term ordering on ground terms. It is well-founded if $>_D$ is well-founded.

Extending term ordering to nonground terms

Interpretation $\varphi(t)$ on a non-ground term t is a function from $X \to D$ to D. This set is naturally equipped with the ordering defined by:

$$f \geq_{X,D} g \text{ iff for all } \rho \in X \to D,\ f(\rho) >_D g(\rho)$$
$$f >_{X,D} g \text{ iff for all } \rho \in X \to D,\ f(\rho) >_D g(\rho)$$

Extending term ordering to nonground terms

Interpretation $\varphi(t)$ on a non-ground term t is a function from $X \to D$ to D. This set is naturally equipped with the ordering defined by:

$$f \geq_{X,D} g \text{ iff for all } \rho \in X \to D,\ f(\rho) >_D g(\rho)$$
$$f >_{X,D} g \text{ iff for all } \rho \in X \to D,\ f(\rho) >_D g(\rho)$$

Definition (Term ordering on all terms)

Let φ be a function mapping each term $t \in \mathcal{T}(\mathcal{F}, X)$ to a function from $X \to D$ to D. The relations $\geq_\varphi$ and $>_\varphi$ generated by φ are defined by:

$$t_1 \geq_\varphi t_2 \text{ iff } \varphi(t_1) \geq_{D,X} \varphi(t_2)$$
$$t_1 >_\varphi t_2 \text{ iff } \varphi(t_1) >_{D,X} \varphi(t_2)$$

Extending term ordering to nonground terms

Interpretation $\varphi(t)$ on a non-ground term t is a function from $X \to D$ to D. This set is naturally equipped with the ordering defined by:

$$f \geq_{X,D} g \text{ iff for all } \rho \in X \to D,\ f(\rho) >_D g(\rho)$$
$$f >_{X,D} g \text{ iff for all } \rho \in X \to D,\ f(\rho) >_D g(\rho)$$

Definition (Term ordering on all terms)

Let φ be a function mapping each term $t \in \mathcal{T}(\mathcal{F}, X)$ to a function from $X \to D$ to D. The relations $\geq_\varphi$ and $>_\varphi$ generated by φ are defined by:

$$t_1 \geq_\varphi t_2 \text{ iff } \varphi(t_1) \geq_{D,X} \varphi(t_2)$$
$$t_1 >_\varphi t_2 \text{ iff } \varphi(t_1) >_{D,X} \varphi(t_2)$$

Lemma (Well-foundedness of $(\geq_\varphi, >_\varphi)$)

$(\geq_\varphi, >_\varphi)$ is a term ordering on non-ground terms. It is well-founded if $>_D$ is well-founded.

Homomorphic interpretations

Definition

For a given domain D, we define a *homomorphic interpretation* φ by providing, for each $f \in \mathcal{F}$ of arity n, a function $[\![f]\!]_\varphi : D^n \to D$, and then inductively on terms: for any $\rho \in X \to D$,

$$\varphi(f(t_1, \ldots, t_n))(\rho) = [\![f]\!]_\varphi(\varphi(t_1)(\rho), \ldots, \varphi(t_n)(\rho))$$
$$\varphi(x)(\rho) = \rho(x)$$

Homomorphic interpretations

Definition

For a given domain D, we define a *homomorphic interpretation* φ by providing, for each $f \in \mathcal{F}$ of arity n, a function $[\![f]\!]_\varphi : D^n \to D$, and then inductively on terms: for any $\rho \in X \to D$,

$$\varphi(f(t_1, \ldots, t_n))(\rho) = [\![f]\!]_\varphi(\varphi(t_1)(\rho), \ldots, \varphi(t_n)(\rho))$$
$$\varphi(x)(\rho) = \rho(x)$$

Lemma (Stability of $(\succeq_\varphi, >_\varphi)$)

If φ is a homomorphic interpretation, then $(\succeq_\varphi, >_\varphi)$ is stable.

Homomorphic interpretations

Definition

For a given domain D, we define a *homomorphic interpretation* φ by providing, for each $f \in \mathcal{F}$ of arity n, a function $[\![f]\!]_\varphi : D^n \to D$, and then inductively on terms: for any $\rho \in X \to D$,

$$\varphi(f(t_1, \ldots, t_n))(\rho) = [\![f]\!]_\varphi(\varphi(t_1)(\rho), \ldots, \varphi(t_n)(\rho))$$
$$\varphi(x)(\rho) = \rho(x)$$

Lemma (Stability of $(\succeq_\varphi, >_\varphi)$)

If φ is a homomorphic interpretation, then $(\succeq_\varphi, >_\varphi)$ is stable.

Lemma (Monotonicity of $(\succeq_\varphi, >_\varphi)$)

For any $f \in \mathcal{F}$ of arity n, and $1 \leq i \leq n$, if for all $d_j \in D$, $1 \leq j \leq n$, $j \neq i$, $[\![f]\!]_\varphi(d_1, \ldots, d_{i-1}, x, d_{i+1}, \ldots, d_n)$ is monotonic (resp. strictly) nondecreasing in x, then $\succeq_\varphi$ (resp. $>_\varphi$) is monotonic with reference to the i-th argument of f.

Choice of domain

- The set of integers - a natural choice.
- Not well-founded in its entirety, we have to consider an appropriate subset

Choice of domain

- The set of integers - a natural choice.
- Not well-founded in its entirety, we have to consider an appropriate subset

Definition (D_μ)

For a given $\mu \in \mathbb{Z}$ let $D_\mu = \{x \in \mathbb{Z} | x \geq \mu\}$.
Interpretations into D_μ are called *arithmetic*, or, should we wish to specify the value of μ, μ-interpretations.

It is easy to see that the usual ordering $>$ is well-founded over each D_μ.

Choice of domain

- The set of integers - a natural choice.
- Not well-founded in its entirety, we have to consider an appropriate subset

Definition (D_μ)

For a given $\mu \in \mathbb{Z}$ let $D_\mu = \{x \in \mathbb{Z} | x \geq \mu\}$.
Interpretations into D_μ are called *arithmetic*, or, should we wish to specify the value of μ, μ-interpretations.

It is easy to see that the usual ordering $>$ is well-founded over each D_μ.

Definition (Polynomial interpretation)

An arithmetic homomorphic interpretation φ defined by functions $[\![f]\!]_\varphi$, $f \in \mathcal{F}$, is called a *polynomial interpretation* if for all f, $[\![f]\!]_\varphi$ is a polynomial function

Requirements for termination

To check whether a given polynomial interpretation is suitable for proving termination of a given TRS using any of the aforementioned criteria, we must be able to check, using positivness-properties, that:

1. each polynomial effectively maps D_μ^n into D_μ – this is fulfilled iff the polynomial $P - \mu$ is nonnegative on D_μ^n;
2. any of the polynomials is weakly and/or strictly increasing in some/all of its arguments – when given a polynomial P with n variables, it is weakly (resp. strictly increasing) in its i-th argument iff the polynomial

$$Q(X_1, \ldots, X_n) = P(X_1, \ldots, X_{i-1}, X_i + 1, X_{i+1}, \ldots, X_n) - P(X_1, \ldots, X_{i-1}, X_i, X_{i+1}, \ldots, X_n) \text{ (resp. } -1)$$

is nonnegative on D_μ^n;

Requirements for termination

3. for any two terms t_1 and t_2, $t_1 \succeq_\varphi t_2$ and/or $t_1 \succ_\varphi t_2$ – when given t_1 and t_2, $\varphi(t_1)$ and $\varphi(t_2)$ can be computed as polynomials P_1 and P_2 over their variables, and then $t_1 \succeq_\varphi t_2$ iff the polynomial $P_1 - P_2$ is nonnegative on D_μ^n, and $t_1 \succ_\varphi t_2$ iff the polynomial $P_1 - P_2 - 1$ is nonnegative on D_μ^n;

Also, if the rewriting is performed modulo an equational theory E, we have to be able to check that

4. $\succeq$ and $\succ$ are compatible with E, in the sense that if $s \succeq t$, $s =_E s'$, and $t =_E t'$, then $s' \succeq t'$, and similarly for $\succ$ – this can be accomplished if for each equation $t \simeq u$ of theory E, we have that $t \succeq_\varphi u$ and $u \succeq_\varphi t$, that is $\varphi(t) - \varphi(u) = 0$;

Approaches to testing positiveness

The positivness problem of polynomial functions – for a given polynomial $P \in \mathbb{Z}[X_1, \ldots, X_n]$, prove that $P(x_1, \ldots, x_n) \geq 0$ for any value $x_i \geq \mu$ – is reducible to the Hilbert's Tenth Problem, which is undecidable.

Approaches to testing positiveness

The positivness problem of polynomial functions – for a given polynomial $P \in \mathbb{Z}[X_1, \ldots, X_n]$, prove that $P(x_1, \ldots, x_n) \geq 0$ for any value $x_i \geq \mu$ – is reducible to the Hilbert's Tenth Problem, which is undecidable.

Proposition (Introducing reals)

The idea is to approximate the problem by checking positivness of polynomial functions for any real values greater than μ of their arguments, which is a decidable problem. However, this has proven to be algorithmically very complex.

The absolute-positivness method, Hong and Jakuš

Definition (μ - absolute positivness)

A polynomial P is said to be μ-absolutely positive iff the polynomial

$$Q(X_1, \ldots, X_n) = P(X_1 + \mu, \ldots, X_n + \mu)$$

has nonnegative coefficients only.

The absolute-positivness method, Hong and Jakuš

Definition (μ - absolute positivness)

A polynomial P is said to be μ-absolutely positive iff the polynomial

$$Q(X_1, \ldots, X_n) = P(X_1 + \mu, \ldots, X_n + \mu)$$

has nonnegative coefficients only.

Lemma (Connecting absolute positivness with positivness)

If P is μ-absolutely positive, then it is nonnegative for all values in D_μ of its variables.

The absolute-positivness method, Hong and Jakuš

Definition (μ - absolute positivness)

A polynomial P is said to be μ-absolutely positive iff the polynomial

$$Q(X_1, \ldots, X_n) = P(X_1 + \mu, \ldots, X_n + \mu)$$

has nonnegative coefficients only.

Lemma (Connecting absolute positivness with positivness)

If P is μ-absolutely positive, then it is nonnegative for all values in D_μ of its variables.

- Calculation of the coefficients of the translated polynomial Q can be quite costly in general.

Computing for $\mu = 0$ is sufficient

Proposition

Let $(\succeq_\varphi, >_\varphi)$ be a term ordering defined by polynomial interpretations with a given μ. Then this ordering can also be defined by some polynomial interpretations with $\mu = 0$.

Computing for $\mu = 0$ is sufficient

Proposition

Let $(\succeq_\varphi, >_\varphi)$ be a term ordering defined by polynomial interpretations with a given μ. Then this ordering can also be defined by some polynomial interpretations with $\mu = 0$.

- Much more efficient to compute the μ-translation of interpretations *once and for all*. When we want to check that $t_1 \succeq_\varphi t_2$, we can compute $\varphi(t_1) - \varphi(t_2)$ and check its positiveness.

Computing for $\mu = 0$ is sufficient

Proposition

Let $(\succeq_\varphi, >_\varphi)$ be a term ordering defined by polynomial interpretations with a given μ. Then this ordering can also be defined by some polynomial interpretations with $\mu = 0$.

- Much more efficient to compute the μ-translation of interpretations *once and for all*. When we want to check that $t_1 \succeq_\varphi t_2$, we can compute $\varphi(t_1) - \varphi(t_2)$ and check its positiveness.
- If we want to search for a polynomial interpretation automatically, *fixing $\mu = 0$ is enough*;

Computing for $\mu = 0$ is sufficient

Proposition

Let $(\succeq_\varphi, >_\varphi)$ be a term ordering defined by polynomial interpretations with a given μ. Then this ordering can also be defined by some polynomial interpretations with $\mu = 0$.

- Much more efficient to compute the μ-translation of interpretations *once and for all*. When we want to check that $t_1 \succeq_\varphi t_2$, we can compute $\varphi(t_1) - \varphi(t_2)$ and check its positiveness.
- If we want to search for a polynomial interpretation automatically, *fixing $\mu = 0$ is enough*;

Lemma (Monotonicity and 0-interpretations)

A polynomial 0-interpretation $P(x_1, \ldots, x_n)$ with nonnegative coefficients is always nondecreasing in each of its arguments. It is strictly increasing in its i-th argument iff there is a monomial ax_i^k, with $a > 0$ and $k > 0$.

The choice of parametric polynomials

- For each symbol of the signature a *parametric polynomial* is chosen
- The coefficients of these polynomials are variables whose values we are searching for
- A bound is fixed on the degree of the polynomials, so that the number of those variables is finite
- The used polynomials follow Steinbach's classification:

The choice of parametric polynomials

- For each symbol of the signature a *parametric polynomial* is chosen
- The coefficients of these polynomials are variables whose values we are searching for
- A bound is fixed on the degree of the polynomials, so that the number of those variables is finite
- The used polynomials follow Steinbach's classification:
  - *linear class*: $P(x_1, \ldots, x_n) = a_1 x_1 + \ldots + a_n x_n + c$;

The choice of parametric polynomials

- For each symbol of the signature a *parametric polynomial* is chosen
- The coefficients of these polynomials are variables whose values we are searching for
- A bound is fixed on the degree of the polynomials, so that the number of those variables is finite
- The used polynomials follow Steinbach's classification:
  - *linear class*: $P(x_1, \ldots, x_n) = a_1 x_1 + \ldots + a_n x_n + c$;
  - *simple class*: $P(x_1, \ldots, x_n) = \sum_{i_j \in \{0,1\}} a_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n}$;

The choice of parametric polynomials

- For each symbol of the signature a *parametric polynomial* is chosen
- The coefficients of these polynomials are variables whose values we are searching for
- A bound is fixed on the degree of the polynomials, so that the number of those variables is finite
- The used polynomials follow Steinbach's classification:
  - *linear class*: $P(x_1, \ldots, x_n) = a_1 x_1 + \ldots + a_n x_n + c$;
  - *simple class*: $P(x_1, \ldots, x_n) = \sum_{i_j \in \{0,1\}} a_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n}$;
  - *simple-mixed class*: $P(x_1, \ldots, x_n) = \sum_{i_j \in \{0,1\}} a_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n} + \sum_{1 \leq i \leq n} b_i x_i^2$;

The choice of parametric polynomials

- For each symbol of the signature a *parametric polynomial* is chosen
- The coefficients of these polynomials are variables whose values we are searching for
- A bound is fixed on the degree of the polynomials, so that the number of those variables is finite
- The used polynomials follow Steinbach's classification:
  - *linear class*: $P(x_1, \ldots, x_n) = a_1 x_1 + \ldots + a_n x_n + c$;
  - *simple class*: $P(x_1, \ldots, x_n) = \sum_{i_j \in \{0,1\}} a_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n}$;
  - *simple-mixed class*: $P(x_1, \ldots, x_n) = \sum_{i_j \in \{0,1\}} a_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n} + \sum_{1 \le i \le n} b_i x_i^2$;
- Additionally, there is one more class of polynomials used:
  - *quadratic class* – $P(x_1, \ldots, x_n) = \sum_{i_j \in \{0,1,2\}} a_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n}$;

Reducing conditions into constraints

Each of the conditions which ensure stability of the defined ordering $(\succeq, \succ)$ have to be translated into constraints on the chosen variables:

Reducing conditions into constraints

Each of the conditions which ensure stability of the defined ordering $(\succeq, \succ)$ have to be translated into constraints on the chosen variables:

- The requirements for $\succeq$ monotonic, $\succ$ and $\succeq$ stable, and $\succ$ well-founded are satisfied as soon as all coefficients are nonnegative;

Reducing conditions into constraints

Each of the conditions which ensure stability of the defined ordering $(\succeq, \succ)$ have to be translated into constraints on the chosen variables:

- The requirements for $\succeq$ monotonic, $\succ$ and $\succeq$ stable, and $\succ$ well-founded are satisfied as soon as all coefficients are nonnegative;
- $\succ$ monotonic w. r. t. the i-th argument of f reduces to $a_i \geq 1$, where a_i is the coefficient of x_i in $[\![f]\!]$;

Reducing conditions into constraints

Each of the conditions which ensure stability of the defined ordering $(\succeq, \succ)$ have to be translated into constraints on the chosen variables:

- The requirements for $\succeq$ monotonic, $\succ$ and $\succeq$ stable, and $\succ$ well-founded are satisfied as soon as all coefficients are nonnegative;
- $\succ$ monotonic w. r. t. the i-th argument of f reduces to $a_i \geq 1$, where a_i is the coefficient of x_i in $[\![f]\!]$;
- $\succeq$ and $\succ$ compatible with an equational theory E reduces to $[\![t]\!] - [\![u]\!] = 0$, for each equation $t \simeq u \in E$;

Reducing conditions into constraints

Each of the conditions which ensure stability of the defined ordering $(\succeq, \succ)$ have to be translated into constraints on the chosen variables:

- The requirements for $\succeq$ monotonic, $\succ$ and $\succeq$ stable, and $\succ$ well-founded are satisfied as soon as all coefficients are nonnegative;
- $\succ$ monotonic w. r. t. the i-th argument of f reduces to $a_i \geq 1$, where a_i is the coefficient of x_i in $[\![f]\!]$;
- $\succeq$ and $\succ$ compatible with an equational theory E reduces to $[\![t]\!] - [\![u]\!] = 0$, for each equation $t \simeq u \in E$;
- $t_1 \succeq t_2$ reduces to $[\![t_1]\!] - [\![t_2]\!] \geq 0$;

Reducing conditions into constraints

Each of the conditions which ensure stability of the defined ordering $(\succeq, \succ)$ have to be translated into constraints on the chosen variables:

- The requirements for $\succeq$ monotonic, $\succ$ and $\succeq$ stable, and $\succ$ well-founded are satisfied as soon as all coefficients are nonnegative;
- $\succ$ monotonic w. r. t. the i-th argument of f reduces to $a_i \geq 1$, where a_i is the coefficient of x_i in $[\![f]\!]$;
- $\succeq$ and $\succ$ compatible with an equational theory E reduces to $[\![t]\!] - [\![u]\!] = 0$, for each equation $t \simeq u \in E$;
- $t_1 \succeq t_2$ reduces to $[\![t_1]\!] - [\![t_2]\!] \geq 0$;
- $t_1 \succ t_2$ reduces to $[\![t_1]\!] - [\![t_2]\!] - 1 \geq 0$;

Reducing conditions into constraints

- For a commutative and associative symbol f, we always choose the interpretation of the form $[\![f]\!](x, y) = axy + b(x + y) + c$, where $b^2 = b + ac$;

Reducing conditions into constraints

- For a commutative and associative symbol f, we always choose the interpretation of the form $[\![f]\!](x, y) = axy + b(x + y) + c$, where $b^2 = b + ac$;
- For a commutative non-associative symbol f, we choose a symmetric interpretation $[\![f]\!](x, y) = [\![f]\!](y, x)$ depending on the class of polynomials used for interpretation:

Reducing conditions into constraints

- For a commutative and associative symbol f, we always choose the interpretation of the form $[\![f]\!](x, y) = axy + b(x + y) + c$, where $b^2 = b + ac$;
- For a commutative non-associative symbol f, we choose a symmetric interpretation $[\![f]\!](x, y) = [\![f]\!](y, x)$ depending on the class of polynomials used for interpretation:
  - for the linear class: $[\![f]\!](x, y) = a(x + y) + b$;

Reducing conditions into constraints

- For a commutative and associative symbol f, we always choose the interpretation of the form $[\![f]\!](x, y) = axy + b(x + y) + c$, where $b^2 = b + ac$;
- For a commutative non-associative symbol f, we choose a symmetric interpretation $[\![f]\!](x, y) = [\![f]\!](y, x)$ depending on the class of polynomials used for interpretation:
  - for the linear class: $[\![f]\!](x, y) = a(x + y) + b$;
  - for the simple class: $[\![f]\!](x, y) = axy + b(x + y) + c$

Reducing conditions into constraints

- For a commutative and associative symbol f, we always choose the interpretation of the form $[\![f]\!](x, y) = axy + b(x + y) + c$, where $b^2 = b + ac$;
- For a commutative non-associative symbol f, we choose a symmetric interpretation $[\![f]\!](x, y) = [\![f]\!](y, x)$ depending on the class of polynomials used for interpretation:
  - for the linear class: $[\![f]\!](x, y) = a(x + y) + b$;
  - for the simple class: $[\![f]\!](x, y) = axy + b(x + y) + c$
  - for the simple-mixed or quadratic class: $[\![f]\!](x, y) = a(x^2 + y^2) + bxy + c(x + y) + d$;

Reducing conditions into constraints

- For a commutative and associative symbol f, we always choose the interpretation of the form $[\![f]\!](x, y) = axy + b(x + y) + c$, where $b^2 = b + ac$;
- For a commutative non-associative symbol f, we choose a symmetric interpretation $[\![f]\!](x, y) = [\![f]\!](y, x)$ depending on the class of polynomials used for interpretation:
  - for the linear class: $[\![f]\!](x, y) = a(x + y) + b$;
  - for the simple class: $[\![f]\!](x, y) = axy + b(x + y) + c$
  - for the simple-mixed or quadratic class: $[\![f]\!](x, y) = a(x^2 + y^2) + bxy + c(x + y) + d$;

In this way, proving termination of a given TRS has been reduced to the problem of solving a set of Diophantine constraints on the coefficients introduces in the parametric interpretations.

An example of a parametric polynomial interpretation

We will use a rewrite system for an endomorphism on a monoid:

$$(x \times y) \times z \to x \times (y \times z)$$
$$f(x) \times f(y) \to f(x \times y)$$
$$f(x) \times (f(y) \times z) \to f(x \times y) \times z$$

An example of a parametric polynomial interpretation

We will use a rewrite system for an endomorphism on a monoid:

$$(x \times y) \times z \to x \times (y \times z)$$
$$f(x) \times f(y) \to f(x \times y)$$
$$f(x) \times (f(y) \times z) \to f(x \times y) \times z$$

For the interpretations of the function symbols, we may choose:

$$[\![f]\!](x) = ax + b$$
$$[\![\times]\!](x, y) = cxy + dx + ey + f.$$

An example of a parametric polynomial interpretation

We will use a rewrite system for an endomorphism on a monoid:

$$(x \times y) \times z \to x \times (y \times z)$$
$$f(x) \times f(y) \to f(x \times y)$$
$$f(x) \times (f(y) \times z) \to f(x \times y) \times z$$

For the interpretations of the function symbols, we may choose:

$$[\![f]\!](x) = ax + b$$
$$[\![\times]\!](x, y) = cxy + dx + ey + f.$$

The constraints obtained from these conditions are the following:

$$a - 1 \geq 0$$
$$d - 1 \geq 0$$
$$e - 1 \geq 0$$
$$cd - ce \geq 0$$
$$d^2 - cf - d \geq 0$$
$$e - e^2 + fc \geq 0$$
$$df - ef - 1 \geq 0$$
$$ca^2 - ca \geq 0$$
$$cab \geq 0$$
$$cb^2 - fa + f + eb + db - b - 1 \geq 0$$
$$c^2a^2 - c^2a \geq 0$$
$$dca^2 - dca \geq 0$$
$$eca - dca + c^2ba \geq 0$$
$$c^2ba \geq 0$$
$$fca - d^2a + dcba + da \geq 0$$
$$e^2 - fca + 2ecb - e + c^2b^2 - cb \geq 0$$
$$dbca \geq 0$$
$$fe - fda + fcb + edb + dcb^2 - 1 \geq 0$$

General idea

- *Put an arbitrary bound on the solution we are looking for* - searching for values of the variables satisfying the constraints in the interval $[0, B]$, where B is some nonnegative integer bound;
- The usual way of solving this problem – a generalization of the DPLL procedure;

General idea

- *Put an arbitrary bound on the solution we are looking for* - searching for values of the variables satisfying the constraints in the interval $[0, B]$, where B is some nonnegative integer bound;
- The usual way of solving this problem – a generalization of the DPLL procedure;
- A *store* – a data structure holding information on possible values for all variables;

General idea

- *Put an arbitrary bound on the solution we are looking for* - searching for values of the variables satisfying the constraints in the interval $[0, B]$, where B is some nonnegative integer bound;
- The usual way of solving this problem – a generalization of the DPLL procedure;
- A *store* – a data structure holding information on possible values for all variables;
- Two most important parts of the algorithm:

General idea

- *Put an arbitrary bound on the solution we are looking for* - searching for values of the variables satisfying the constraints in the interval $[0, B]$, where B is some nonnegative integer bound;
- The usual way of solving this problem – a generalization of the DPLL procedure;
- A *store* – a data structure holding information on possible values for all variables;
- Two most important parts of the algorithm:
  - *Constraint propagation* – a procedure which, given a store, performs logical deductions in order to obtain a smaller store;

General idea

- *Put an arbitrary bound on the solution we are looking for* - searching for values of the variables satisfying the constraints in the interval $[0, B]$, where B is some nonnegative integer bound;
- The usual way of solving this problem – a generalization of the DPLL procedure;
- A *store* – a data structure holding information on possible values for all variables;
- Two most important parts of the algorithm:
  - *Constraint propagation* – a procedure which, given a store, performs logical deductions in order to obtain a smaller store;
  - *Nondeterministic branching* which explores all possible values of variables, with various heuristics;

Pseudo-code of the algorithm

```
Solve(C:constraint, B:integer) =
  let S = {0 → x.min, B → x.max | x ∈ Var(C)} in Branch(S, C)

Branch(S:store, C:constraint) =
  let S = propagate(S, C) in
      if for each x∈S, x.min = x.max
         then if isSolution(S, C)
              then output
              else throw noSolution
         else
             let x = chooseVar(S) in
             try
                let S1 = {S with S.x.min → x.max} in Branch(S1, C)
             catch noSolution →
             let S2 = {S with S.x.min + 1 → x.min} in Branch(S2, C)
```

Pseudo-code of the algorithm

```
Solve(C:constraint, B:integer) =
  let S = {0 → x.min, B → x.max | x ∈ Var(C)} in Branch(S, C)

Branch(S:store, C:constraint) =
  let S = propagate(S, C) in
      if for each x∈S, x.min = x.max
         then if isSolution(S, C)
              then output
              else throw noSolution
         else
             let x = chooseVar(S) in
             try
                let S1 = {S with S.x.min → x.max} in Branch(S1, C)
             catch noSolution →
             let S2 = {S with S.x.min + 1 → x.min} in Branch(S2, C)
```

Pseudo-code of the algorithm

```
Solve(C:constraint, B:integer) =
  let S = {0 → x.min, B → x.max | x ∈ Var(C)} in Branch(S, C)

Branch(S:store, C:constraint) =
  let S = propagate(S, C) in
      if for each x∈S, x.min = x.max
         then if isSolution(S, C)
              then output
              else throw noSolution
         else
             let x = chooseVar(S) in
             try
                let S1 = {S with S.x.min → x.max} in Branch(S1, C)
             catch noSolution →
             let S2 = {S with S.x.min + 1 → x.min} in Branch(S2, C)
```

Choosing variables

After experimentation with various heuristics, the best one for the selection of variables in `chooseVar(S)` appears to be the following one:

Choosing variables

After experimentation with various heuristics, the best one for the selection of variables in `chooseVar(S)` appears to be the following one:

- choose the variable with the smallest value of `min`;

Choosing variables

After experimentation with various heuristics, the best one for the selection of variables in `chooseVar(S)` appears to be the following one:

- choose the variable with the smallest value of `min`;
- of all such variables, choose the variable with the largest value of `max`;

Choosing variables

After experimentation with various heuristics, the best one for the selection of variables in `chooseVar(S)` appears to be the following one:

- choose the variable with the smallest value of `min`;
- of all such variables, choose the variable with the largest value of `max`;
- of all such variables, choose the one which occurs most often in the constraints;

An example

Let's consider the following Diophantine constraints, which we will attempt to solve for x, y in the interval $[0, 100]$:

$$(1)\ \ 2x + y \leq 12 \qquad (2)\ \ xy = 15$$

First we build the initial store, and from there progress toward the solution:

An example

Let's consider the following Diophantine constraints, which we will attempt to solve for x, y in the interval $[0, 100]$:

$$(1)\ \ 2x + y \leq 12 \qquad (2)\ \ xy = 15$$

First we build the initial store, and from there progress toward the solution:

x	0	100
y	0	100

Pr, (1), (2) →

x	2	4
y	4	8

Br, x=2 →

x	2	2
y	4	8

Pr, (1) →

x	2	2
y	8	7

Br, x=3 →

x	3	4
y	4	8

Pr, (1), (2) →

x	3	4
y	4	5

Br, x=3 →

x	3	3
y	4	5

Pr, (1), (2) →

x	3	3
y	5	5

Finite domain constraints and optimizations

The existing constraints were transformed into *finite domain constraints* of the form $x \in [e_1, e_2]$, for some expressions e_1, e_2, with appropriate changes made to the algorithm. Also, various further optimizations were implemented, such as:

- Simplification of the polynomials
- Square and product abstraction
  - $S, C \Rightarrow S \cup \{z.min = 0,\ z.max = (x.max)^2\}, C[x^2/z] \cup \{z = x^2\}$
  - $S, C \Rightarrow S \cup \{z.min = 0,\ z.max = x.max \times y.max\}, C[xy/z] \cup \{z = xy\}$
- Minimization of the number of introduced variables

Implementation

- Implementation of the described algorithm in the CiME rewrite tool
- Possibility of selecting the class of polynomial interpretations to be used and the bound on coefficients
- Possibility of selecting the optimization techniques which will be used:
  - FD(0) – no squares or products abstracted
  - FD(1) – all squares and products abstracted
  - FD(2) – only squares and products occuring at least twice are abstracted

Results

- The implemented method allowed for solving thousands of constraints, over hundreds of variables, within a few seconds
- Different abstraction methods tested over a large collection of TRSs from the TPDB:

		FD(0)	FD(1)	FD(2)
solved TRSs	number	304	315	314
	percentage	53.0%	54.9%	54.7%
	average time	1.44s	0.89s	0.47%
unsolved TRSs	number	270	259	260
	percentage	47.0%	45.1%	45.3%
	average time	26.47s	27.02s	26.77%

The analysis of the results has shown that:

- Policies FD(1) and FD(2) are superior to FD(0), but equivalent with each other, when it comes to the number of problems solved
- The policy FD(2) yields the quickest results

References

Contejean, E., Marché, Tomas, A. P. and Urbain, X.:
Mechanically proving termination using polynomial interpretations.
Journal of automated Reasoning, 32(4), pp. 315–355, 2006.