~~Matrix Interpretations~~ $\mathbb{N}$-weighted Finite Automata with Polyhedral Constraints

Johannes Waldmann (HTWK Leipzig)

June 23, 2015

Automated Analysis of Termination and Complexity

- . . . for the computation model of (term or string) *rewriting*
- applications: (first order) functional programs, RNA transformations
- we use $(\mathbb{N}, +, \cdot)$-weighted automata, defines an evaluation algebra over $\mathbb{N}^{Q(A)}$
- determine transitions (weights) of automaton by solving a constraint system
- *new feature: restrict the domain* to $D \subset \mathbb{N}^{Q(A)}$

String Rewriting

- alphabet Σ, set of rules $R \subseteq \Sigma^* \times \Sigma^*$,
- defines relation (one-step R-rewriting) $u \to_R v$
  $\iff \exists (l, r) \in R, p, q \in \Sigma^* : u = plq \land prq = v$.
- R is syntax (program),
  $\to_R^*$ is semantics (computation)
- examples: $R_{\text{bubble sort}} = \{ba \to ab\}$,
  $R_{\text{exponentiation}} = \{ab \to baa\}$, $R_? = \{aa \to aba\}$
- related: semi-Thue-systems, Markov algorithms, Turing machines, formal grammars

Termination, and How to Prove It

- Def: R is strongly normalizing (or: (uniformly) terminating), written $\mathrm{SN}(R)$ iff there is no infinite $\to_R$-chain.
- Examples: $\mathrm{SN}(\{ab \to ba\})$, $\neg\,\mathrm{SN}(\{ab \to b^2a^2\})$
- SN is undecidable (cf. TM halting problem)

methods to prove termination

- syntactical (e.g., consider overlaps between parts of rules)
- (this talk) *semantical* (assign some meaning to the objects that are being rewritten)

Goal: *automate* the methods (and their *certification*)

Automated Termination Analysis

Why? Want tools that help in …

- analysis of source/machine code (in IDE/in OS)
- completion of equational specifications
- theorem proving (check that induction is well-founded)

How to measure progress? Compete!

- annual Termination Competitions `http://termination-portal.org/wiki/Termination_Competition_2015`
- termination provers run on benchmarks (last year, $2.7 \cdot 10^4$ "job pairs", $4 \cdot 10^6$ seconds CPU)

Interpretations

- Def: partial order $(D, >)$ is *well-founded*: has no infinite $>$-chains
- Def: interpretation $i : \Sigma^* \to (D, >)$ is *compatible* with rewrite system R if $u \to_R v \Rightarrow i(u) > i(v)$.
- R admits compatible interpretation into some wf domain $\iff$ $\mathrm{SN}(R)$
- note: "$\Leftarrow$" is trivial, take $i = \mathrm{id}$ and $D = (\Sigma^*, \to_R^+)$
- example: for $R_{\text{bubblesort}} = \{ba \to ab\}$, count inversions: $i(w) = |\{(j,k) \mid j < k \wedge w_j > w_k\}|$ then $u \to_R v \Rightarrow i(u) - 1 = i(v)$
- example: for $\{aa \to aba\} \dots ?$

Monotone Algebras

Σ-algebra A on wf $(D, >)$

- $\epsilon_A \in D$, and for each $f \in \Sigma$, a function $f_A : D \to D$
- A defines an interpretation $i_A : \Sigma^* \to D$
- Def: A is *monotone* iff
  $\forall f \in \Sigma : \forall x, y \in D : x > y \Rightarrow f_A(x) > f_A(y)$
- Def: A is *compatible with R* if
  $\forall (l, r) \in R, \forall x \in D : l_A(x) > r_A(x)$.
- Thm: R admits a compatible monotone algebra over a well-founded domain $\iff$ SN(R).
  note: "$\Leftarrow$" is still trivial
- "number of inversions" is not an algebra (over $\mathbb{N}$)

Linear Algebra

domain $(\mathbb{N}^d, >)$

- with $\vec{x} > \vec{y} \; := \; x_1 > y_1 \wedge x_2 \geq y_2 \wedge \dots \wedge x_d \geq y_d$ is well-founded

example: 1st comp. counts number of *aa* factors:
$[a](x_1, x_2) = (x_1 + x_2, 1), [b](x_1, x_2) = (x_1, 0)$

- is monotone:
  all coeffs. ≥ 0, coeff. of x_1 in 1st comp. is > 0
- is compatible with $R = \{aa \to aba\}$:
  $[aa](x_1, x_2) = (x_1 + x_2 + \boxed{1}, 1)$,
  $[aba](x_1, x_2) = (x_1 + x_2, 0)$

this algebra is the algebra of a weighted automaton

Algebras of Weighted Automata

- D-weighted FA A
  - alphabet Σ, states Q
  - initial weight vector $I : D^Q$
  - transitions $T : \Sigma \to D^{Q\times Q}$
  - final weigth vector $F : D^Q$
- its weight function: $A : \Sigma^* \to D : w \mapsto F \cdot T(w) \cdot I$
- its algebra (with carrier D^Q): given by T and I
- D could be any semiring (because we need properties of matrix multiplication)
- here, we restrict to $D = (\mathbb{N}, +\cdot, 0, 1)$,
  I picks the initial state, F picks the final state

Algebras from Automata, Example

automaton A:
$$\begin{tikzcd} {} \arrow[r, "1"] & 1 \arrow[loop above, "\Sigma : 1"] \arrow[r, "a : 1"] & 2 \arrow[r, "a : 1"] & 3 \arrow[loop above, "\Sigma : 1"] \arrow[r, "1"] & {} \end{tikzcd}$$

transitions

$$T(a) = \begin{pmatrix} \boxed{1} & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \boxed{1} \end{pmatrix}, T(b) = \begin{pmatrix} \boxed{1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \boxed{1} \end{pmatrix},$$

compute

$$T(aa) = \begin{pmatrix} 1 & 1 & \boxed{1} \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} > \begin{pmatrix} 1 & 1 & \boxed{0} \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} = T(aba)$$

WFA and Rewriting

goal: the algebra of a WFA A is well-founded, monotone, and compatible with R
$\Rightarrow$ A is a certificate of termination of R

- wf: use domain $(\mathbb{N}^d, >)$ as defined earlier
- monotone (on the left): for each a, $T(a)_{1,1} \geq 1$
- compatible with R:
  $T(l)_{1,d} > T(r)_{1,d}$,
  and for each $(l, r) \in R$: $T(l) \geq T(r)$ (point-wise),
  and monotone on the right: $T(a)_{d,d} \geq 1$.

From Strings to Terms

- interpret k-ary letter f by
  $f_A : (\vec{x_1}, \ldots, \vec{x_k}) \mapsto \vec{F_0} + \sum_i F_i \cdot \vec{x_i}$
  where F_0 is vector, $F_1, \ldots$ are square matrices,
- this is a restricted form of multi-linear functions, closed w.r.t. composition (needed for interpretation of terms with variables)
- $\forall i \geq 1 : (F_i)_{11} \geq 1$ implies *monotonicity*,
- for rule $l \to r$, compute
  $[l](\vec{x_1}, \ldots) = L_0 + \sum_i L_i \cdot \vec{x_i}, [r](\vec{x_1}, \ldots) = R_0 + \ldots$
  then $\forall i \geq 0 : L_i \geq R_i$ (component-wise)
  and $(L_0)_1 > (R_0)_1$ implies *compatibility*

From Strings to Terms … and Back

- for 1-ary, this means $f_A(\vec{x}) = F_0 + F_1\vec{x}$

  equivalently, $f_A\begin{pmatrix}\vec{x}\\1\end{pmatrix} = \begin{pmatrix}F_1 & F_0\\0\cdots0 & 1\end{pmatrix}\cdot\begin{pmatrix}\vec{x}\\1\end{pmatrix}$
- previous interpretation (automaton)

  $[]_A : a \mapsto \begin{pmatrix}\boxed{1} & 1 & 0\\0 & 0 & 1\\0 & 0 & \boxed{1}\end{pmatrix}, b \mapsto \begin{pmatrix}\boxed{1} & 0 & 0\\0 & 0 & 0\\0 & 0 & \boxed{1}\end{pmatrix},$

  will be written

  $[a]_A(\vec{x}) = \begin{pmatrix}0\\1\end{pmatrix} + \begin{pmatrix}1 & 1\\0 & 0\end{pmatrix}\vec{x},$

  $[b]_A(\vec{x}) = \begin{pmatrix}0\\0\end{pmatrix} + \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}\vec{x}$

Constraints for Unknown Automata

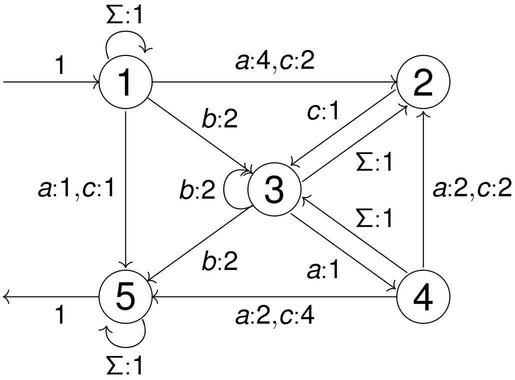
- "monotone, and compatible with R"
  is a *constraint system* for the transition matrices
- e.g., from $R = \{ba \to ab\}$ with
  $[a](x) = a_0 + a_1 x, [b](x) = b_0 + b_1 x$ obtain $a_0 \geq 0, a_1 \geq 1, b_0 \geq 0, b_1 \geq 1, b_1 a_0 + b_0 > a_1 b_0 + a_0$
- write constraint system in suitable form

```
(declare-fun P () Int) (declare-fun Q () Int)
(declare-fun R () Int) (declare-fun S () Int)
(assert (and (< 0 P) (<= 0 Q) (< 0 R) (<= 0 S
(assert (> (+ (* P S) Q) (+ (* R Q) S)))
```

  constraint solver searches satisfying assignment
- this is (now) a standard method in automated termination

Matrix Interpretation Success Story

Termination of $\{a^2 \to bc, b^2 \to ac, c^2 \to ab\}$ follows from this automaton:



(Hofbauer, Waldmann, 2006)

Polyhedral Domains

(this is the point of the RTA'15 paper)

- standard method uses domain $(\mathbb{N}^d, >)$, now restrict to some subset $D \subset \mathbb{N}^d$ defined by a conjunction of linear inequalities
- D contains the weight vectors reachable by A
- behaviour of transitions of A outside D is ignored
- relaxed proof obligations for compatibility $\forall x \in D : [l](x) > [r](x)$
- additional proof obligations $D \neq \emptyset, \forall a \in \Sigma : [a](D) \subseteq D$
- get *more* and *different* termination proofs

Polyhedral Constraints, Example

Prove termination of $R = \{fg \to ff, gf \to gg\}$.
Use domain $D = \{(x_1, x_2, x_3) \in \mathbb{N}^3 \mid x_3 \geq x_2 + 1\}$.

$$
\begin{aligned}
[f](x_1, x_2, x_3) &= (x_1 + 2x_2 + 1, 0, x_3 + 1) \\
[g](x_1, x_2, x_3) &= (x_1 \qquad\qquad , x_3, x_3 + 1)
\end{aligned}
$$

$$
\begin{aligned}
[fg](x) &= (x_1 + \qquad\quad 2x_3 + \boxed{1}, 0, x_3 + 2), \\
[ff](x) &= (x_1 + \boxed{2}x_2 + \qquad \boxed{2}, 0, x_3 + 2).
\end{aligned}
$$

Now $\forall x \in D : [fg](x) > [ff](x)$, despite $\boxed{2}$.
$x_1 + 2x_3 + 1 \geq x_1 + (2x_2 + \boxed{2}) + \boxed{1} > x_1 + 2x_2 + 2$

Interpret. with Polyhedral Constraints

A polyhedrally constrained matrix interpret. contains:

- the interpretation, $f_A(x_1, \dots) = F_0 + \sum F_i x_i$
- the domain, given by $C_A \in \mathbb{Q}^{c \times d}, B_A \in \mathbb{Q}^{c \times 1}$, as $D = \{x \mid x \geq 0, Cx + B \geq 0\} \subseteq \mathbb{N}^d$

In the example, $d = 3, c = 1, C = (0, -1, 1), B = -1$.

to use it for termination of rewriting, we show:

- domain is non-empty,
- interpretation respects the domain,
- interpretation is compatible with rules.

for each of these, we use *certificates*

Polyhedral Constraints: Domains

Def: *A respects the domain* if $f_A : D^k \to D$.
This is certified by giving

- for each letter f, with interpretation $f_A(x_1, \dots) = F_0 + \sum F_i x_i$,
- matrices $W_1, \dots W_k \in \mathbb{Q}_{\geq 0}^{c \times c}$ with $CF_0 + B \geq (\sum_i W_i)B$, $\forall 1 \leq i \leq k : CF_i \geq W_i C$

example: $D = \{(x_1, x_2, x_3) \in \mathbb{N}^3 \mid x_3 \geq x_2 + 1\}$,
$[f](x_1, x_2, x_3) = (x_1 + 2x_2 + 1, 0, x_3 + 1)$
take $W_1 = 0$

Polyhedral Constraints: Compatibility

Compatibility of A w.r.t. rule $(l \to r)$
with $|\operatorname{Var}(l) \cup \operatorname{Var}(r)| = k$
where $([l]_A - [r]_A)(x_1, \ldots, x_k) = \Delta_0 + \sum_i \Delta_i x_i$,

is certified by matrices $U_1, \ldots, U_k \in \mathbb{Q}_+^{d \times c}$,
such that $\forall i : \Delta_i \geq U_i C$ and $\Delta_0 > \sum_i U_i B$

example: $D = \{\vec{x} \in \mathbb{N}^3 \mid -x_2 + x_3 - 1 \geq 0\}$,
$[f](\vec{x}) = (x_1 + 2x_2 + 1, 0, x_3 + 1)$,
$[g](\vec{x}) = (x_1, x_3, x_3 + 1)$,
$[fg - ff](\vec{x}) = (-2x_2 + 2x_3 - 1, 0, 0)$
take $U_1 = (2, 0, 0)^T$.

Polyhedral Constraints: Combined

to prove termination of rewriting system *R*, determine

- matrix interpretation (weighted automaton)
- polyhedral domain (linear inequalities)

as solution of a constraint system for validity of certificates for

- non-emptiness of the domain
- respecting the domain
- compatibility with rules

implemented in termination prover Matchbox2015.

Completeness of Certificates

Thm: automaton respects domain, is R-compatible $\iff$ certificates exist.

- Correctness ("$\Leftarrow$") is easily verified.
- Completeness ("$\Rightarrow$") follows from (inhomogenous) Farkas' Lemma.

The Lemma (in one of many versions) says

- A linear inequalitiy I is implied by a system S of linear inequalities
- $\iff$ $I \geq$ some positive linear combination of S.

Derivational Complexity

- motivation: (automated) analysis of complexity of programs
- derivation height of a term, w.r.t. $\to$: $\operatorname{dh}(\to, s) = \sup\{k \mid \exists t : s \to^k t\}$
- derivational complexity of $\to$: $\operatorname{dc}(\to) = n \mapsto \max\{\operatorname{dh}(\to, s) \mid |s| \le n\}$
- example: $\operatorname{dc}(\to_{\{ba \to ab\}}) \in \Theta(n \mapsto n^2)$
- "derivational complexity" is an (extra) category of Termination Competitions

Deriv. Complexity and Interpretations

- complexity of matrix interpretation (using matrices from some set $\mathcal{M}$) $\mathrm{dc}(\mathcal{M}) = n \mapsto \max\{\|M\| : M \in \mathcal{M}^{\leq n}\}$
- Thm: if $\mathcal{M}$ is finite and *upper triangular* (0 below main diagonal, 0 or 1 on main diag.), then $\mathrm{dc}(\mathcal{M})$ is polynomial
- polyhedral domain restriction is orthogonal to this, but sometimes helpful
- ex. $R = \{fg \to ff, gf \to gg\}$: given automaton is upper triangular, this proves $\mathrm{dc}(R)$ quadratic, this was known, but by different (more complicated) method (root labelling)

Results, Discussion, Announcement

- main result: method is correct, implementation.
- auxiliary results, see paper
- challenge: improve implementation (improve constraint solver, better bit-blasting)
- challenge: automated proof of quadratic derivational complexity of $\{a^2 \to cb, b^2 \to ca, c^2 \to ba\}$
- open: extend to other semirings, e.g., arctic.
- for more on rewriting and termination: *8th Intl. School on Rewriting, Leipzig, August 10-14*

  `http://nfa.imn.htwk-leipzig.de/ISR2015/`