On the Construction of Matchbound Certificates

Johannes Waldmann (HTWK Leipzig)

TERMGRAPH 2016

Matchbound Certificates

A matchbound certificate
for a string rewriting system R over alphabet Σ
is a directed graph G with edge labels from $\Sigma \times \mathbb{N}$
representing an $\mathbb{F}$-weighted automaton A (details on next slides) such that

- $\forall c \in \Sigma : A(p_0, c, p_0) \neq 0_{\mathbb{F}}$
- $\forall p, q \in V(G), (l, r) \in R : A(p, l, q) <_{\mathbb{F}} A(p, r, q)$

basic steps for construction:

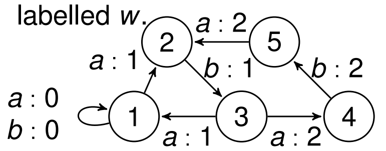
- compute weights for lhs $A(p, l, q)$, rhs $A(p, r, q)$
- if $\not<_{\mathbb{F}}$, add edges, to increase weight of rhs

Certificate exists $\Rightarrow$ R is (cycle) terminating.

Fuzzy Automata

The "fuzzy" semiring (not a standard name) $\mathbb{F}$
$(\{-\infty\} \cup \mathbb{N} \cup \{+\infty\}, \max, -\infty (= 0_{\mathbb{F}}), \min, +\infty (= 1_{\mathbb{F}}))$
$x <_{\mathbb{F}} y :\iff (x = 0_{\mathbb{F}} = y) \vee (x < y)$ $\mathbb{F}$-weighted automaton A:

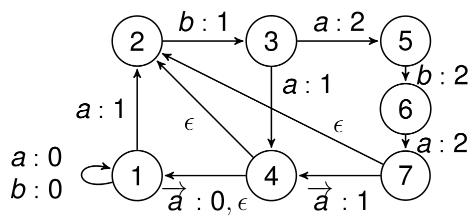
- directed edges, labelled by $\Sigma \times \mathbb{F}$ (letter, weight)
- weight of a path: min of weights of edges
- label of a path: concatenation of letters at edges
- $A(p, w, q)$: max of weights of $p - q$ paths labelled w.



is matchbound certificate for $\{aa \to aba\}$

Algorithms f. Certificate Construction

- Hofbauer, W. 2003? : complete, slow
- Zantema 2004? : incomplete, fast
- Endrullis 2006 : complete, fast, uses ϵ trans. and formal inverses $\overrightarrow{c_h} c_{\geq h} \to \epsilon$, $c_{\geq h} \overleftarrow{c_h} \to \epsilon$



Certificate Construction

1. [TRANS] if there are p, q with $(A_\epsilon \cdot A_\epsilon)(p, q) \neq 0_{\mathbb{F}}$, add $p \overset{\epsilon}{\to} q$.
2. [INV] if there are c, p, q with $A_\epsilon(c\overleftarrow{c})(p, q) \neq 0_{\mathbb{F}}$, or $A_\epsilon(\overrightarrow{c}c)(p, q) \neq 0_{\mathbb{F}}$, add $p \overset{\epsilon}{\to} q$.
3. [REWRITE] if there is $(l, r) \in R$ and (p, q) such that $A_\epsilon(l)(p, q) \not<_{\mathbb{F}} A_\epsilon(r)(p, q)$, then:
   - let $p' \overset{c:h}{\to} q'$ with $c \in \Sigma, h \in \mathbb{F}$ be a transition of minimal height on a maximal l-labelled (p, q)-Path, such that $l = sct$ for $s, t \in \Sigma^*$,
   - then add a path from p' to q' over fresh states only that consists of a sequence of edges labelled by $\overrightarrow{s}$ with height h, r with weight $h + 1$, $\overleftarrow{t}$ with height h.

An Online Algorithm

- "if there is some path, then add some path"
- use data structure that
  - contains weight of paths
  - allows cheap ddition of edges

A old graph, Δ new edges, $A' = A + \Delta$
in semiring of $\mathbb{F}$ matrices:
$A'(w_1 \cdot w_2) = (A(w_1) + \Delta(w_1)) \cdot (A(w_2) + \Delta(w_2)) =$
$A(w_1 \cdot w_2) + A'(w_1) \cdot \Delta(w_2) + \Delta(w_1) \cdot A'(w_2)$
compute bottom-up, along a *multiplication chain*
for the (finite) set of strings of interest
want multiplication that is fast if one factor is small.

Representing Weighted Relations

$R \subseteq (P \times Q) \to S$

- dense matrix representation wastes space
- sparse representation: must allow quick access to sets of (nonzero) predecessors and successors of each node.

```
data Rel p q s =
  Rel { fore :: Map p (Map q s)
      , back :: Map q (Map p s) }

times :: Rel p q s -> Rel q r s -> Rel p r s
times r1 r2 = M.foldl ...
  $ M.intersectionWith ... (back r1) (fore r2)
```


Extra Information for $\mathbb{F}$ (Inverses)

- a formal left inverse of c: $\overrightarrow{c_h} \cdot c_{\geq h} \to \epsilon$
- in the automaton: $A(p, \overrightarrow{c}, q) \leq A(r, c, s) \wedge A_\epsilon(q, r) = 1_{\mathbb{F}} \Rightarrow A\epsilon(p, s) = 1_{\mathbb{F}}$

represent this as semiring multiplication

```
data F' = MinusInf | Fin Nat | PlusInf
     | LeftInv Nat | RightInv Nat

times (LeftInv h) (Fin h')
  = if h <= h' then PlusInf else MinusInf
```

is this really a semiring? Does not matter too much, we don't need arbitrary products.

Extra Information for $\mathbb{F}$ (Location)

- algorithm computes weight $A(p, w, q) \in \mathbb{F}$.
- rule says: if $A(p, l, q) \not\leq_{\mathbb{F}} A(p, r, q)$, then add path ... from p' to q', where $p' \overset{c:h}{\to} q'$ is a minimal edge on a maximal $p - q$ path

```
data I = I { weight :: F
    , from  :: Q, to :: Q -- ^ minimal edge
    , offset :: Int, total :: Int }

plus i j = if weight i <= weight j then i else j
times i j = if weight i >= weight j
  then i { total = total i + total j }
  else j { total = total i + total j
         , offset = total i + offset j }
```


Implementation, Performance

- `https://gitlab.imn.htwk-leipzig.de/waldmann/pure-matchbox` in Haskell, uses `Data.IntMap` (Patricia trees) from `containers` library
- “killer example” `SRS/secret06/jambox1`: RFC match bound 12, certificate with 43.495 nodes, in 8 sec.
  can you beat this? using your graph rewriting tool/library
- performance on TPDB (SRS standard and cycle termination) see web site, contains links to starexec jobs

Ongoing Work: Streams of Automata

- current main program is imperative (it updates the automaton)
- alternative formulation should be possible: certificate automaton as the limit of a stream
- defined by a productive system of equations, in a stream algebra
- stream $[A_0, A_1, \dots]$ represented by $[A_0, \Delta_1, \dots]$ where $A_k + \Delta_{k+1} = A_{k+1}$ and Δ is ultimately 0
- Problems:
  - productivity
  - priority of rules (TRANS, INV $>$ REWRITE)