Theory I: The Essentials

Word equations

- what can be done (and how)
- what cannot be done (in general)
- what is unknown (open)

Matthew Hague, **Artur Jeż**, Anthony Widjaja Lin, Philipp Ruemmer

Theoretical perspective

String solving = solving Equations + constraints

Word equations (with constraints)

Complexity (decidability): depends on constraint types

- regular constraints: PSPACE
- CFG constraints, letter-counting: undecidable
- linear length constraints: open

String equations and inequations, ex. theory

Equation: $U = V$, where U, V are sequences of letters (Σ) and variables
Inequation: $U \neq V$

If $U \neq V$ then

- U is longer: $U = VaU'$ for some a (letter), U' (variable) or
- V is longer: $V = UaV'$ for some a (letter), V' (variable) or
- first difference $U = WaU'$, $V = WbV'$ for $a \neq b$ (letters) and W, U', V' (new variables)

Nondeterministic reduction.

Existential Theory algorithm: remove alternative (guess), remove inequations (guess)
Left with system of eqautions.

Lentin/Plotkin/Siekman algorithm; Nielsen's transform; Matyasevich

$x\ldots = y\ldots$

- $a \neq b$ (contradiction)
- $x \sqsubset y$ (prefix) $\Rightarrow \boxed{y \leftarrow xy}$
- $y \sqsubset x$ (prefix) $\Rightarrow \boxed{x \leftarrow yx}$
- $x = \varepsilon$ or $y = \varepsilon$

Choose, substitute and delete leading symbols

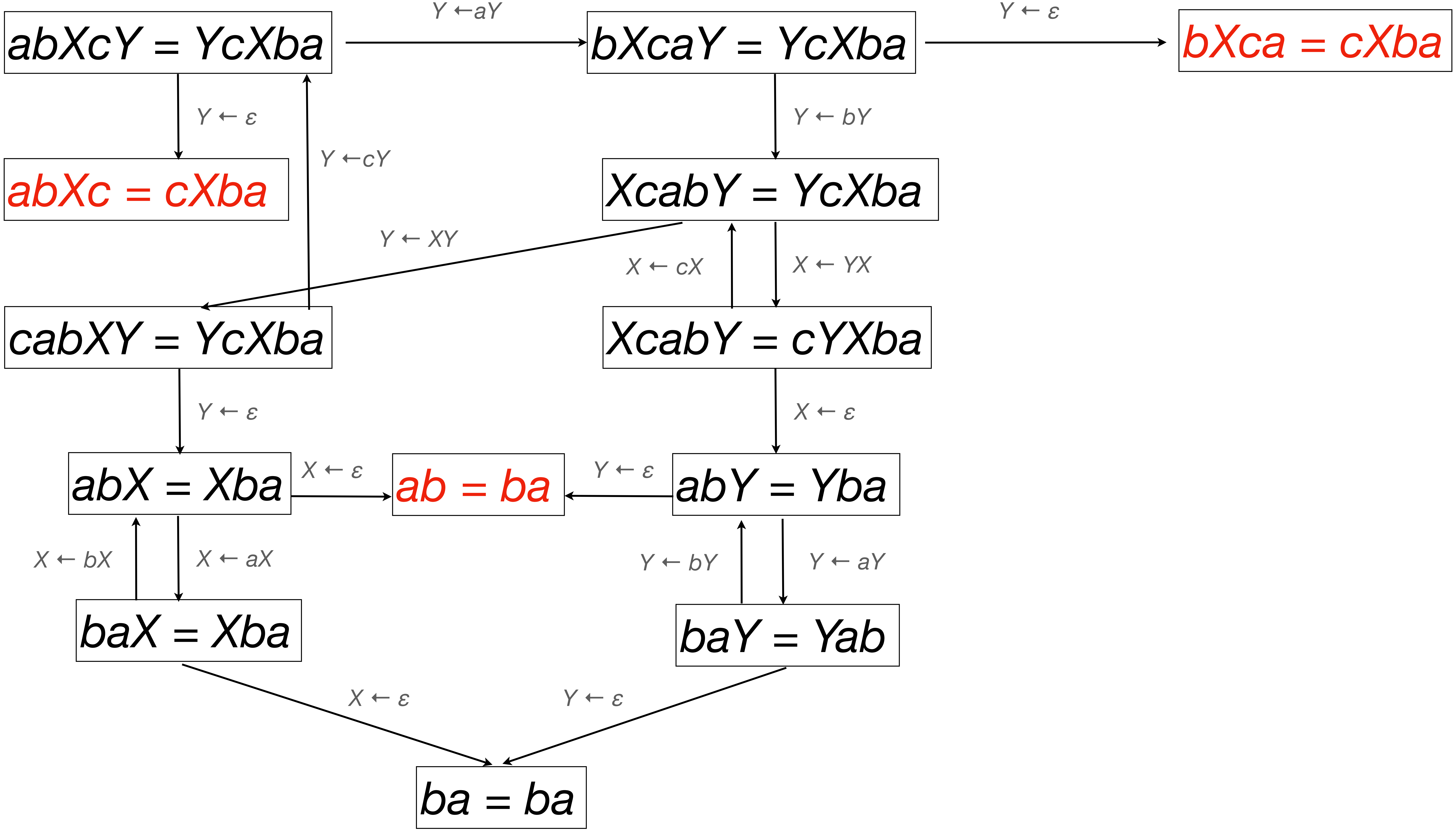
Used in practice (esp.: restricted instances)

Sound

Satisfiable $\Rightarrow$ complete

Unsatisfiable $\Rightarrow$? contradictions or explore whole search space

Plotkin's algorithm is complete on quadratic equations



Makanin's algorithm

First decidability in the general case

Generalizes the Plotkin algorithm (keeps much more info)

Extends to regular constraints

- complex
- high complexity
- proof: difficult string combinatorics
- difficult to generalize

Restricted classes

Better algorithms for restricted classes

- quadratic equations (each variable occurs twice)
- two variables
- one variable
- …

Undecidable constraints

- CFG constraints: intersection of CFGs
- letter-counting constraints (linear):

  $$|X|_a = 1 + 2|X|_b$$

  encodes Diophantine equations
- …

Encoding in string equations is difficult

Length constraints

$$|X| = 1 + 2|Y|$$

Big open problem in the area

The known algorithms change/spoil lengths

The undecidability of letter-counting does not translate

Compression enters the stage

Plandowski '98 PSPACE

Generelizes reasonably (regular constraints, reversal, …)

Jeż '12: simpler algorithm and analysis:

- good on its own
- ⇒ very robust: generalizes very well

We give the algoritm and proof (no constraints)

Some notation and basics

$X, Y, Z \ldots$ variables

a, b, c: letters Σ: alphabet

S: substitution (of variables by strings) $S(X)$

S: extends to sequences of letters and variables

S: solution of $U = V$ when $S(U) = S(V)$ (solution string)

S: length-minimal solution: for all solutions S'

$$|S(U)| \leq |S'(U)|$$

Theorem: Length minimal solution is at most doubly exponential

Conjecture: at most exponential $\Rightarrow$ in NP (widely believed)

$$a\ a\ a\ b\ a\ b\ c\ a\ b\ a\ b\ b\ a\ b\ c\ b\ a$$

$$a\ a\ a\ b\ a\ b\ c\ a\ b\ a\ b\ b\ a\ b\ c\ b\ a$$

$$a\ a\ a\ b\ a\ b\ c\ a\ b\ a\ b\ b\ a\ b\ c\ b\ a$$

$$a\ a\ a\ b\ a\ b\ c\ a\ b\ a\ b\ b\ a\ b\ c\ b\ a$$

$$a_3 \quad b\ a\ b\ c\ a\ b\ a\ b\ b\ a\ b\ c\ b\ a$$

$$a_3 \quad b\ a\ b\ c\ a\ b\ a\ b\ b\ a\ b\ c\ b\ a$$

$$a_3 \quad b \ a \ b \ c \ a \ b \ a \quad b_2 \ a \ b \ c \ b \ a$$

$$a_3 \quad b \ a \ b \ c \ a \ b \ a \quad b_2 \ a \ b \ c \ b \ a$$

$$a_3 \quad b \quad d \quad c \quad d \quad a \quad b_2 \quad d \quad c \; b \; a$$

$$a_3 \quad b \quad d \quad c \quad d \quad a \quad b_2 \quad d \quad c \; b \; a$$

$$a_3 \quad b \quad d \quad c \quad d \quad a \quad b_2 \quad d \quad c \quad e$$

$$a_3 \quad b \quad d \quad c \quad d \quad a \quad b_2 \quad d \quad c \quad e$$

$$a_3 \quad b \quad d \quad c \quad d \quad a \quad b_2 \quad d \quad c \quad e$$

$$a_3 \quad b \quad d \quad c \quad d \quad a \quad b_2 \quad d \quad c \quad e$$

Intuition: recompression

- Think of new letters as nonterminals of a grammar
- We build a grammar for both strings, bottom-up.
- Everything is compressed in the same way!

Idea

while $U \notin \Sigma$ and $V \notin \Sigma$ do

$L \leftarrow$ letters from $S(U) = S(V)$

for $ab \in L^2$ or $a \in L$ do

replace all occurrences of ab in $S(U)$ and $S(V)$

(or replace all occurrences of runs of a)

How to do it for an equation?

Idea at work

$XbaYb = baaababbbab$ has a solution $S(X) = baaa$, $S(Y) = bba$

We want to replace pair ba by a new letter c. Then

$XbaY\ b{=}baaababbbab$ for $S(X) = baaa\ S(Y) = bba$

$Xc\ \ Y\ b{=}c\ \ aac\ \ bc\ \ b$ for $S(X) = c\ \ aa\ S(Y) = bc$

And what about replacing ab by d?

$XbaYb = baaababbbab$ has a solution $S(X) = baaa$, $S(Y) = bba$

There is a problem with `crossing pairs'. We will fix it!

Pair types

Occurrence of *ab* in a solution string (so for a fixed solution) is

- explicit it comes from U or V;
- implicit comes solely from $S(X)$;
- crossing in other case.

ab is crossing if it has a crossing occurrence, non-crossing otherwise.

X *baa* Y *b*	$= baaabaabbab$	$S(X) = baaa$ $S(Y) = bba$
baaa baa bba b	$= baaabaabbab$	explicit
baaa baa bba b	$= baaabaabbab$	implicit
baaa baa bba b	$= baaabaabbab$	crossing

Compression of non-crossing pairs

PairComp (a, b)

let $c \in \Sigma$ be an unused letter

replace each explicit ab in U and V by c

Lemma: PairComp (a, b) is sound

If ab is noncrossing: it is complete.

Nondeterminism: assumption that ab is noncrossing

Completeness

define $S'(X)$: $S(X)$ with every ab replaced with c

Lemma: $S'(U')$ is $S(U)$ with every ab replaced; similarly $S'(V')$

explicit pairs   replaced explicitly

implicit pairs   replaced implicitly (in the substitution)

crossing pairs   there are none

$X \quad baa \quad Y \ \ b{=}baaabaabbab$ $S(X) = baaa \ S(Y) = bba$

$baaa \ baa \ bba \ b{=}baaabaabbab$

$c \ aa \ \ c \ a \ b \ c \ \ b{=} \ c \ aa \ c \ ab \ c \ b$

$X \quad c \ a \quad Y \quad b{=} \ c \ aa \ c \ ab \ c \ b$ $S'(X) = caa \ S'(Y) = bc$

Soundness

define $S(X)$: $S'(X)$ with every c replaced with ab

Lemma: $S(U)$ is $S'(U')$ with every c replaced by ab; similarly $S(V)$

explicit c — replaced explicitly

implicit c — replaced implicitly (in the substitution)

$X \quad c\ a \quad Y \quad b = c\ aa\ c\ ab\ c\ b$ $\qquad S'(X) = caa\ S'(Y) = bc$

$c\ aa \quad c\ a\ b\ c \quad b = c\ aa\ c\ ab\ c\ b$

$baaa\ baa\ bba\ b = baaabaabbab$

$X \quad baa \quad Y \quad b = baaabaabbab$ $\qquad S(X) = baaa\ S(Y) = bba$

Dealing with crossing pairs

ab is a crossing pair ⇔ there is X s.t. $\boxed{S(X) = bw}$ and aX occurs in $U = V$ (or symmetric).

Uncrossing(a, b)

for variable X do

if first letter of $S(X)$ is b then

replace each occurrence of X by bX \\Pop; S changes accordingly

if $S(X)$ is empty then remove X from the equation

perform symmetrically for the last letter and a

Lemma After uncrossing ab is no longer crossing ⇒ we can compress it

Uncrossing: example

$X \quad baa \quad Y \; b \quad = baaabaabbab \qquad S(X) = baaa \quad S(Y) = bba$

$baaa\; baa\; bba\; b \quad = baaabaabbab$

$baaa\; baa\; bba\; b \quad = baaabaabbab$

$bXa\; baa\; bYa \;\; b \quad = baaabaabbab \qquad S'(X) = aa \quad S'(Y) = b$

Maximal blocks

Maximal block of *a: when* a^k occurs in $S(U) = S(V)$ and cannot be extended.

Block occurrence can be explicit, implicit or crossing.

Letter a is crossing (has a crossing block) if there is a crossing block of a.

$X \quad baaa \quad Y \quad b \quad = baabaaabbb$ $\qquad S(X) = baab \ S(Y) = bb$

$baab \ baaa \ bb \ b \quad = baabaaabbb$

Lemma If a^k is a maximal block in a length-minimal solution of $U = V$ then $k \leq 2^{c|UV|}$.

Blocks compression

When a has no crossing block

for all maximal blocks a^k of a and $k > 1$ do

let $a_k \in \Sigma$ be an unused letter

replace each explicit maximal a^k in $U = V$ by a_k

Lemma BlockComp(a) is sound.
If a is noncrossing then it is complete

$X \; baaa \; Y \; b = baabaaabbb$ $\qquad$ $S(X) = baab$ $S(Y) = bb$

$baab \; baaa \; bb \; b = baabaaabbb$

$b \, a_2 b \; b \, a_3 \; bb \; b = b \, a_2 b \, a_3 \, bbb$

$X \; b \, a_3 \; Y \; b = b \, a_2 b \, a_3 \, bbb$ $\qquad$ $S'(X) = ba_2b$ $S'(Y) = bb$

Crossing *a*-blocks?

As for pairs? Popping a single *a:* not enough

pop whole *a*-prefix and *a*-suffix:

$S(X) = a^l\ w\ a^r$: change it to $S(X) = w$

for variable X do

replace each occurrence of X by $a^l\ X\ a^r$ \\ a^l, a^r: the *a*-prefix *a*-suffix of $S(X)$

if S(X) is empty then

remove X from the equation

Lemma: After uncrossing *a* is no longer crossing.

The algorithm

while $U \notin \Sigma$ and $V \notin \Sigma$ do

$L \leftarrow$ letters from $S(U) = S(V)$

choose $ab \in L^2$ or $a \in L$ \\\\ very flexible about the order

if it is crossing then

uncross it

compress it

Soundness

If the new equation has a solution, then also the original one had.

Just roll back the changes.

$X \quad c\,a \quad Y \quad b = c\,aa\,c\,ab\,c\,b$ $\qquad S'(X) = caa \; S'(Y) = bc$

$c\,a\,a\,c\,a\,b\,c \quad b = c\,aa\,c\,ab\,c\,b$

$baa\;abaa\;bba\;b = baaabaabbab$

$X \quad baa \quad Y \quad b = baaabaabbab$ $\qquad S(X) = baaa \; S(Y) = bba$

Completeness

Equation has the solution, then
for some nondeterministic choices the new equation has a corresponding one.

Make the choices according to the solution.

What about termination?

Termination

We show that

- we stay in $O(n^2)$ space. (can be $O(n)$)
- after each operation the length-minimal solution shortens.

Terminate on positive instances.
Explore whole space for negative instances.

Lemma: Each compression decreases the length of the length-minimal solution

We perform the compression on the solution word:
there is a shorter solution
the shortest may be even shorter

Strategy

Lemma: Compression of a non-crossing pair/block decreases equation's size.

Something is compressed in the equation.

Strategy:

- If there is something non-crossing: compress it.
- If there are only crossing: choose one that minimises the equation.

Lemma: There are at most $2n$ different crossing pairs and blocks. (For a fixed solution)

Each is associated with a side of an occurrence of a variable.

Lemma: Uncrossing introduces at most $2n$ letters to the equation.

Each variable pops left and right one letter
for a-blocks: it is compressed immediately afterwards.

Lemma: There is always a choice to be $\leq 8n^2$.

There are $m \leq 8n^2$ letters in the equation and $k \leq 2n$ different crossing blocks/pairs.

Some covers $\geq m/k$ letters.

Its compression removes $\geq (m/k)/2 = m/2k$ letters and introduces $2n$ letters.

We are left with at most

$$m - m/2k + 2n = (1 - 1/2k) \cdot m + 2n \leq (1 - 1/4n) \cdot 8n^2 + 2n = 8n^2$$

Remarks

- representation, not combinatorial properties.
- robust:

  different variant of compressions

  order of operations

  …

- bottom-up: difficult in practice.
- heavy non-determinism.
- spoils lengths

Regular constraints

Regular constraints: which formalism?

User likes: $X \in r \wedge X \in r' \wedge X \notin r''$
r described in some way (DFA, NFA, RE, …)

Theory likes: transition matrices (or transition monoid)

- Boolean matrices for words
- $M(w)_{pq} = 1 \Leftrightarrow$ we can go from q to p by w.
- Concatenation: Boolean matrix multiplication
- Constraint: give $M(X)$, require $M(S(X)) = M(X)$

Can translate (at some cost).

Keep $M(X)$ in the algorithm, compute $M(c)$ for new letters

Tasks: