Negated String Containment is Decidable

Vojtěch Havlena[1] Michal Hečko[1] Lukáš Holík[1] Ondřej Lengál[1]

[1]Brno University of Technology, Czech Republic

MFCS'25

Negated string containment
and its semantics

NEGATED STRING CONTAINMENT

Input: A formula

$$\varphi \triangleq \neg \mathit{contains}(\mathcal{N}, \mathcal{H}) \land \bigwedge_{X \in \mathbb{X}} X \in L_X$$

with $\mathcal{N}, \mathcal{H} \in (\Sigma \cup \mathbb{X})^*$ and L_X is regular for every variable X.

Question: Find a morphism $\sigma : \mathbb{X} \to \Sigma^*$ such that

- $\sigma(X) \in L_X$ for every variable X, and
- $\sigma(\mathcal{N})$ is not a factor $\sigma(\mathcal{H})$.

We call $\mathcal{N}$ and $\mathcal{H}$ the $\mathcal{N}$eedle and $\mathcal{H}$aystack, respectively.

Negated string containment

Examples

Given

$$\varphi \triangleq \neg contains(XabY, YababX) \quad \wedge \quad X \in a^* \quad \wedge \quad Y \in b^*$$

We have

- $\{X \mapsto a, Y \mapsto b\} \models \varphi$
- $\{X \mapsto \varepsilon, Y \mapsto \varepsilon\} \not\models \varphi$

Negated string containment
Examples

Given

$$\varphi \triangleq \neg contains(XabY, YababX) \quad \wedge \quad X \in a^* \quad \wedge \quad Y \in b^*$$

We have

- $\{X \mapsto a, Y \mapsto b\} \models \varphi$
- $\{X \mapsto \varepsilon, Y \mapsto \varepsilon\} \not\models \varphi$

Alternatively,

$$\varphi' \triangleq \neg contains(XabY, YababX) \quad \wedge \quad X \in (ab)^* \quad \wedge \quad Y \in (ab)^*$$

has no models.

Motivation

Symbolic execution and SMT solving

```
# Check if pwd can be writen as concat(w, ..., w) for some word w
pwd = input()  # pwd='abab'
pwd2 = concat(pwd, pwd) # pwd2='abababab'
pwd2_inner = pwd2[1:-1] # pwd2_inner='bababa'
if is_substring(pwd, pwd2_inner):
    report_weak_password()
else:
    proceed()
```

What value of `pwd` causes `proceed()` to be called?

$$P_2 = P_1 \circ P_1 \quad \wedge \quad P_2 = U \circ P_3 \circ V \wedge U, V \in \Sigma \quad \wedge \quad \neg contains(P_1, P_3)$$

Challenge

The $\neg contains(\mathcal{N}, \mathcal{H})$ formula can be equivalently expressed as an $\exists\forall$-quantified disequation

$$\exists \vec{X}(\neg contains(\mathcal{N}, \mathcal{H})) \quad \Leftrightarrow \quad \exists \vec{X}\, \forall P, S(P \circ \mathcal{N} \circ S \neq \mathcal{H})$$

[1] V. G. Durnev. “Positive formulas in free semigroups”. In: *Siberian Mathematical Journal* 15.5 (1974).

Challenge

The $\neg contains(\mathcal{N}, \mathcal{H})$ formula can be equivalently expressed as an $\exists\forall$-quantified disequation

$$\exists \vec{X}(\neg contains(\mathcal{N}, \mathcal{H})) \quad \Leftrightarrow \quad \exists \vec{X}\, \forall P, S(P \circ \mathcal{N} \circ S \neq \mathcal{H})$$

Quantifiers are notoriously difficult, quickly leading to undecidability

- already the $\exists^1\forall^1\exists^3$-fragment is known to be undecidable[1]

[1] V. G. Durnev. "Positive formulas in free semigroups". In: *Siberian Mathematical Journal* 15.5 (1974).

Preliminaries

Connection between automata and Presburger arithmetic

Theorem (Modified Parikh theorem)

Let $\mathcal{A}$ be an NFA. There is an effectively constructable Presburger arithmetic (PA) formula φ_{Parikh} of size polynomial in $|\mathcal{A}|$ such that

1. *any model $\sigma \models \varphi_{Parikh}$ corresponds to an accepting run ρ of $\mathcal{A}$, and*
2. *$\sigma(q \xrightarrow{a} r)$ is the number of times the transition $q \xrightarrow{a} r$ is taken by ρ.*

Preliminaries

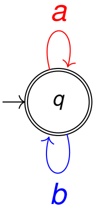
Connection between automata and Presburger arithmetic

Theorem (Modified Parikh theorem)

Let $\mathcal{A}$ be an NFA. There is an effectively constructable Presburger arithmetic (PA) formula φ_{Parikh} of size polynomial in $|\mathcal{A}|$ such that

1. *any model $\sigma \models \varphi_{Parikh}$ corresponds to an accepting run ρ of $\mathcal{A}$, and*
2. *$\sigma(q \xrightarrow{a} r)$ is the number of times the transition $q \xrightarrow{a} r$ is taken by ρ.*

We can reason about automaton runs in decidable Presburger arithmetic. However, commutativity prevents precise reasoning.



$$\sigma = \{q \xrightarrow{a} q \mapsto 1, q \xrightarrow{b} q \mapsto 1\}$$

$$\sigma \models \varphi_{Parikh}$$

$$\sigma \rightsquigarrow w_1 = ab \in L$$

$$\sigma \rightsquigarrow w_2 = ba \in L$$

Preliminaries

Flat languages

Regular language L is flat if it has the form:

$$L = \bigcup_{1 \leq i \leq N} u_{i,0}(w_{i,1})^* \cdots (w_{i,k_i})^* u_{i,k_i}$$

where $u_{i,j}, w_{i,k} \in \Sigma^*$.

Preliminaries

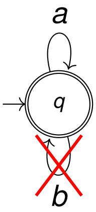
Flat languages

Regular language L is flat if it has the form:

$$L = \bigcup_{1 \leq i \leq N} u_{i,0}(w_{i,1})^* \cdots (w_{i,k_i})^* u_{i,k_i}$$

where $u_{i,j}, w_{i,k} \in \Sigma^*$.

A flat language is a regular language for which every model of its φ_{Parikh} corresponds to exactly one $w \in L$.



Preliminaries

Decision procedure for flat ¬*contains*

If all variables are flat, we can reason precisely about variable assignments in PA, i.e., we can construct an equisatisfiable quantified PA formula.

Theorem

Let φ be a formula

$$\varphi \triangleq \neg \mathit{contains}(\mathcal{N}, \mathcal{H}) \wedge \bigwedge_{X \in \mathbb{X}} X \in L_X$$

such that, for every variable X, the language L_X is flat. Then satisfiability of φ is decidable[a]*.*

---

[a] Yu-Fang Chen et al. “A Uniform Framework for Handling Position Constraints in String Solving”. In: PLDI (2025).

Narrowing down the question

When is $\neg contains(\mathcal{N}, \mathcal{H})$ easy?

$$\varphi = \neg contains(\mathcal{N}, \mathcal{H}) \wedge \bigwedge_{X \in \mathbb{X}} X \in L_X$$

Solving φ is **easy** when:

- we can find $\sigma : \mathbb{X} \to \Sigma^*$ such that $|\sigma(\mathcal{N})| > |\sigma(\mathcal{H})|$ (using φ_{Parikh}),
- all variables are flat, or
- $\mathcal{N}$ is a literal.

Narrowing down the question

When is $\neg contains(\mathcal{N}, \mathcal{H})$ easy?

$$\varphi = \neg contains(\mathcal{N}, \mathcal{H}) \wedge \bigwedge_{X \in \mathbb{X}} X \in L_X$$

Solving φ is **easy** when:

- we can find $\sigma : \mathbb{X} \to \Sigma^*$ such that $|\sigma(\mathcal{N})| > |\sigma(\mathcal{H})|$ (using φ_{Parikh}),
- all variables are flat, or
- $\mathcal{N}$ is a literal.

Solving φ is **hard** when every non-flat variable $X \in \mathbb{X}$ satisfies:

1. X occurs both in $\mathcal{H}$ and $\mathcal{N}$, or
2. X occurs only in $\mathcal{H}$.

Step 1: Normalization

Restricting the structure of regular languages

Step 1: Normalization

Input: A formula

$$\varphi \triangleq \neg contains(\mathcal{N}, \mathcal{H}) \land \bigwedge_{X \in \mathbb{X}} X \in L_X$$

Output: An equisatisfiable disjunction

$$\bigvee_{i \in I} (\neg contains(\mathcal{N}_i, \mathcal{H}_i) \land \bigvee_{X \in \mathbb{X}} X \in L_{X,i})$$

such that

- every flat variable X has a language w_X^* for some $w_X \in \Sigma^*$,
- every non-flat variable Y has a language S_Y^* for some $S_Y \subseteq \Sigma^*$.

$$\neg \mathit{contains}(\mathcal{N}, \mathcal{H}) \land \bigwedge_{X \in \mathbb{X}} X \in L_X$$

How to handle non-flat variables occurring both in $\mathcal{H}$ and $\mathcal{N}$?

Dealing with situations when Y is on both sides

Our goal

Ultimately, we obtain the following lemma.

Lemma

Let $Y \in \mathbb{X}$ be a non-flat variable, $\square \notin \Sigma$ be a fresh symbol and

$$\varphi = \neg contains(u_0 Y u_1 \cdots Y u_n, v_0 Y v_1 \cdots Y v_m) \wedge \bigwedge_{X \in \mathbb{X}} X \in L_X.$$

Then φ is equisatisfiable to φ', where

$$\varphi' = \neg contains(u_0 \square u_1 \cdots \square u_n, v_0 \square v_1 \cdots \square v_m) \wedge \bigwedge_{X \in \mathbb{X} \setminus \{Y\}} X \in L_X.$$

Dealing with situations when Y is on both sides

Proof sketch, direction from φ' to φ

Let

$$\varphi = \neg contains(u_0 Y u_1 \cdots Y u_n, v_0 Y v_1 \cdots Y v_m) \wedge \bigwedge_{X \in \mathbb{X}} X \in L_X$$

$$\varphi' = \neg contains(\underbrace{u_0 \square u_1 \cdots \square u_n}_{\mathcal{N}'}, \underbrace{v_0 \square v_1 \cdots \square v_m}_{\mathcal{H}'}) \wedge \bigwedge_{X \in \mathbb{X} \setminus \{Y\}} X \in L_X.$$

Assume that we have an assignment $\sigma' : \mathbb{X} \setminus \{Y\} \to \Sigma^*$, such that $\sigma' \models \varphi'$.

Dealing with situations when Y is on both sides

Proof sketch, direction from φ' to φ

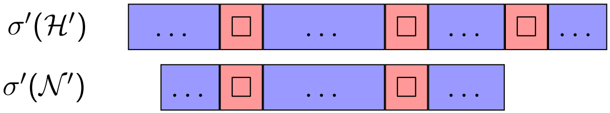
Let

$$\varphi = \neg contains(u_0\, Y u_1 \cdots Y u_n, v_0\, Y v_1 \cdots Y v_m) \wedge \bigwedge_{X \in \mathbb{X}} X \in L_X$$

$$\varphi' = \neg contains(\underbrace{u_0 \square u_1 \cdots \square u_n}_{\mathcal{N}'}, \underbrace{v_0 \square v_1 \cdots \square v_m}_{\mathcal{H}'}) \wedge \bigwedge_{X \in \mathbb{X} \setminus \{Y\}} X \in L_X.$$

Assume that we have an assignment $\sigma' : \mathbb{X} \setminus \{Y\} \to \Sigma^*$, such that $\sigma' \models \varphi'$.

Intuitively, σ' is interesting only when $\square$ in $\sigma'(\mathcal{N}')$ is above some $\square$ in $\sigma'(\mathcal{H}')$.



Dealing with situations when Y is on both sides

Proof sketch, direction from φ' to φ

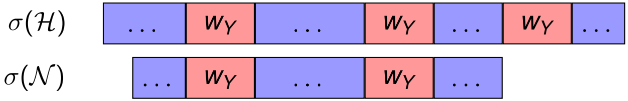
Let $w_Y \in L_Y$, and let $\sigma \triangleq \sigma' \triangleleft \{ Y \mapsto w_Y \}$.

Dealing with situations when Y is on both sides

Proof sketch, direction from φ' to φ

Let $w_Y \in L_Y$, and let $\sigma \triangleq \sigma' \triangleleft \{ Y \mapsto w_Y \}$.

Since $\sigma' \models \neg contains(\mathcal{N}', \mathcal{H}')$ we know that



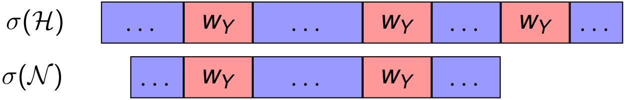
contains a conflict.

Dealing with situations when Y is on both sides

Proof sketch, direction from φ' to φ

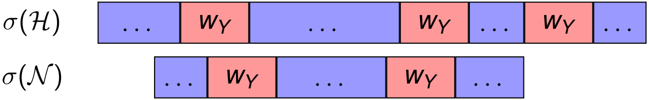
Let $w_Y \in L_Y$, and let $\sigma \triangleq \sigma' \triangleleft \{ Y \mapsto w_Y \}$.

Since $\sigma' \models \neg \mathit{contains}(\mathcal{N}', \mathcal{H}')$ we know that



contains a conflict.

Therefore, if $\sigma \not\models \neg \mathit{contains}(\mathcal{N}, \mathcal{H})$, we cannot have every w_Y in $\sigma(\mathcal{N})$ under some w_Y in $\sigma(\mathcal{H})$.

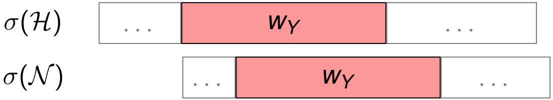


Dealing with situations when Y is on both sides

Proof sketch, direction from φ' to φ

If $\sigma \not\models \neg \mathit{contains}(\mathcal{N}, \mathcal{H})$, we can force some w_Y from $\sigma(\mathcal{N})$ to overlap with w_Y from $\sigma(\mathcal{H})$

- by picking long enough w_Y

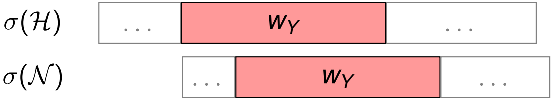


Dealing with situations when Y is on both sides

Proof sketch, direction from φ' to φ

If $\sigma \not\models \neg contains(\mathcal{N}, \mathcal{H})$, we can force some w_Y from $\sigma(\mathcal{N})$ to overlap with w_Y from $\sigma(\mathcal{H})$

- by picking long enough w_Y



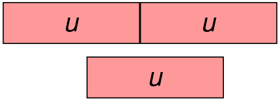
All that is needed is to come up with a special w_Y that cannot have conflict-free overlaps (of sufficient size) with itself, which would allow us to *always* construct a model σ from σ'.

Enter combinatorics on words

How to choose w_Y with the desired properties

A word u is called *primitive* if $u \notin w^*$ for any word $w \neq u$.

Primitive words have cool properties, e.g., if $uu = pus$, then either $p = \varepsilon$ or $s = \varepsilon$.
Graphically, the following is not possible.



Applying combinatorics on words

Proof sketch, direction from φ' to φ

Thanks to our normalization, we have $\{u, v\}^* \subseteq L_Y$ with $u, v \notin w^*$ for any word w.

Applying combinatorics on words

Proof sketch, direction from φ' to φ

Thanks to our normalization, we have $\{u, v\}^* \subseteq L_Y$ with $u, v \notin w^*$ for any word w.

Let us define α and β as

$$\alpha \triangleq u^2 u^k v^2 \quad \in L_Y$$
$$\beta \triangleq u^2 v^l v^2 \quad \in L_Y$$

for $k = \mathrm{lcm}(|u|, |v|)/|u|$ and $l = \mathrm{lcm}(|u|, |v|)/|v|$.

Lemma

Both α and β are primitive.[a]

[a] R. C. Lyndon and M. P. Schützenberger. "The equation $a^M = b^N c^P$ in a free group.". In: *Michigan Mathematical Journal* 9.4 (1962).

Applying combinatorics on words

Proof sketch, direction from φ' to φ

Thanks to our normalization, we have $\{u, v\}^* \subseteq L_Y$ with $u, v \notin w^*$ for any word w.

Let us define α and β as

$$\alpha \triangleq u^2 u^k v^2 \quad \in L_Y$$
$$\beta \triangleq u^2 v^l v^2 \quad \in L_Y$$

for $k = \mathrm{lcm}(|u|, |v|)/|u|$ and $l = \mathrm{lcm}(|u|, |v|)/|v|$.

Lemma

Both α and β are primitive.[a]

[a] R. C. Lyndon and M. P. Schützenberger. “The equation $a^M = b^N c^P$ in a free group.”. In: *Michigan Mathematical Journal* 9.4 (1962).

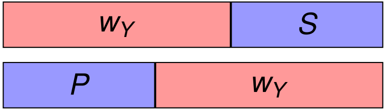
Finally, the word

$$w_Y \triangleq \alpha^M \beta^M \alpha^M \beta^M \alpha^{2M} \beta^{2M} \quad \in L_Y$$

prevents large self-overlaps, where $M = \lceil |M_{\mathrm{Lit}}| / |\alpha| \rceil$ and M_{Lit} is the longest literal in φ.

Lemma

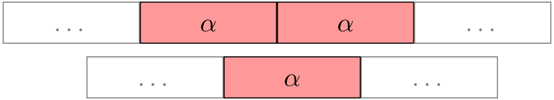
The equation $w_Y S = P w_Y$ has no solutions with $|S| \leq |w_Y| - (M+1)|\alpha|$.



Details on our choice of W_Y

Proof sketch, direction from φ' to φ

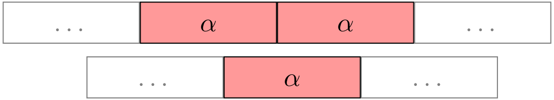
To show that w_Y truly has the desired properties we first observe that whenever we consider a long enough overlap, we have α^2 above α (or similarly for β).



Details on our choice of W_Y

Proof sketch, direction from φ' to φ

To show that w_Y truly has the desired properties we first observe that whenever we consider a long enough overlap, we have α^2 above α (or similarly for β).

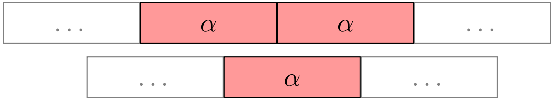


Recall, that since α is primitive, the equation $\alpha^2 = P\alpha S$ has only solutions with $P = \varepsilon$ or $S = \varepsilon$. Therefore, we need to consider overlaps of w_Y with w_Y only with certain granularity.

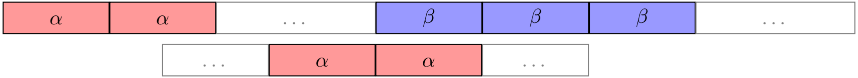
Details on our choice of W_Y

Proof sketch, direction from φ' to φ

To show that w_Y truly has the desired properties we first observe that whenever we consider a long enough overlap, we have α^2 above α (or similarly for β).



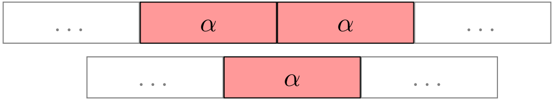
Recall, that since α is primitive, the equation $\alpha^2 = P\alpha S$ has only solutions with $P = \varepsilon$ or $S = \varepsilon$. Therefore, we need to consider overlaps of w_Y with w_Y only with certain granularity.



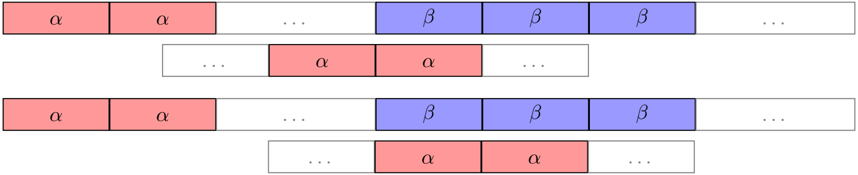
Details on our choice of W_Y

Proof sketch, direction from φ' to φ

To show that w_Y truly has the desired properties we first observe that whenever we consider a long enough overlap, we have α^2 above α (or similarly for β).



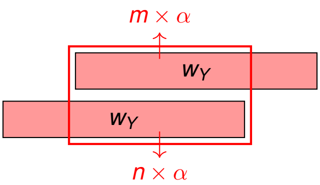
Recall, that since α is primitive, the equation $\alpha^2 = P\alpha S$ has only solutions with $P = \varepsilon$ or $S = \varepsilon$. Therefore, we need to consider overlaps of w_Y with w_Y only with certain granularity.



Details on our choice of W_Y

Proof sketch, direction from φ' to φ

For any of such remaining 'granular' overlaps we directly show that whenever we consider an overlap of w_Y with itself, there is a different number of α's in the overlapping portions of w_Y from $\sigma(\mathcal{N})$ and $\sigma(\mathcal{H})$[2].



[2] except in one case

Dealing with situations when Y is on both sides

STEP 2. REMOVING VARIABLES OCCURRING ON BOTH SIDES

Input: A formula

$$\varphi = \neg contains(\mathcal{N}, \mathcal{H}) \wedge \bigwedge_{X \in \mathbb{X}} X \in L_X$$

Output: An equisatisfiable formula

$$\varphi' = \neg contains(\mathcal{N}', \mathcal{H}') \wedge \bigwedge_{X \in \mathbb{X}} X \in L_X$$

such that every non-flat variable Y occurring both in $\mathcal{N}$ and $\mathcal{H}$ has been replaced by a corresponding $\square_Y$, yielding $\mathcal{N}'$ and $\mathcal{H}'$. I.e. we iteratively replace suitable variables by fresh symbols.

$$\neg contains(\mathcal{N}, \mathcal{H}) \land \bigwedge_{X \in \mathbb{X}} X \in L_X$$

How to handle non-flat variables occurring only in $\mathcal{H}$?

Non-flat variables occurring only in $\mathcal{H}$

Our goal

Lemma

Let $Y \in \mathbb{X}$ be a non-flat variable, and let

$$\varphi = \neg contains(\mathcal{N}, v_0 Y v_1 \cdots Y v_n) \wedge \bigwedge_{X \in \mathbb{X}} X \in L_X.$$

There is a formula φ' equisatisfiable to φ such that

$$\varphi' = \neg contains(\mathcal{N}, v_0 Y v_1 \cdots Y v_n) \wedge Y \in L'_Y \wedge \bigwedge_{X \in \mathbb{X} \setminus \{Y\}} X \in L_X$$

with $L'_Y \subseteq L_Y$ being a flat language.

Non-flat variables occurring only in $\mathcal{H}$

Naive approach

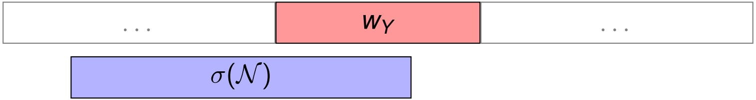
Again, assume a partial assignment $\sigma' \colon \mathbb{X} \setminus \{Y\} \to \Sigma^*$.

Non-flat variables occurring only in $\mathcal{H}$

Naive approach

Again, assume a partial assignment $\sigma' \colon \mathbb{X} \setminus \{Y\} \to \Sigma^*$.

A naive approach would be to enumerate $w \in L_Y$, and check whether $\sigma \triangleq \sigma' \triangleleft \{Y \mapsto w\}$ is a model.

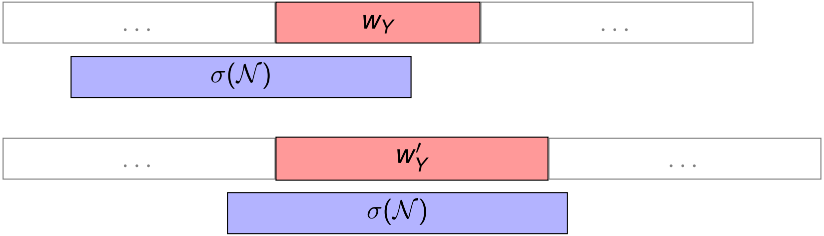


Non-flat variables occurring only in $\mathcal{H}$

Naive approach

Again, assume a partial assignment $\sigma' : \mathbb{X} \setminus \{Y\} \to \Sigma^*$.

A naive approach would be to enumerate $w \in L_Y$, and check whether $\sigma \triangleq \sigma' \triangleleft \{Y \mapsto w\}$ is a model.

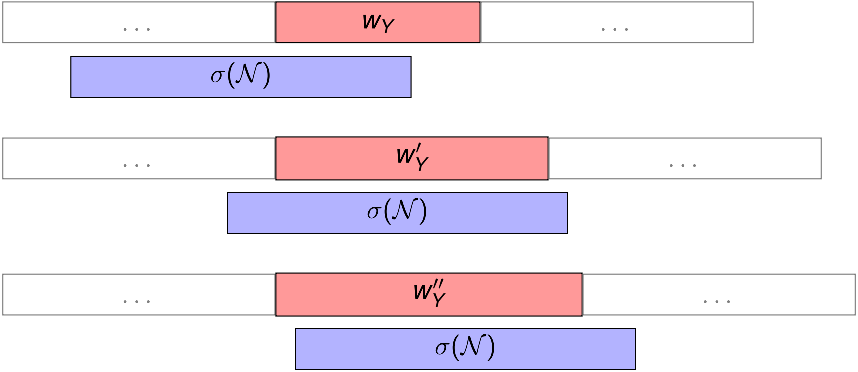


Non-flat variables occurring only in $\mathcal{H}$

Naive approach

Again, assume a partial assignment $\sigma' : \mathbb{X} \setminus \{Y\} \to \Sigma^*$.

A naive approach would be to enumerate $w \in L_Y$, and check whether $\sigma \triangleq \sigma' \triangleleft \{Y \mapsto w\}$ is a model.



Non-flat variables occurring only in $\mathcal{H}$

It is problematic to know why σ fails to be a model.

Non-flat variables occurring only in $\mathcal{H}$

It is problematic to know why σ fails to be a model.

It would be much easier to solve modified formulae

$$\varphi_{Pref} \triangleq \neg contains(\mathcal{N}[Y/Y_p\#], \mathcal{H}[Y/Y_p\#]),$$
$$\varphi_{Suf} \triangleq \neg contains(\mathcal{N}[Y/\#Y_s], \mathcal{H}[Y/\#Y_s]).$$

where $\#$ is a fresh separator symbol and Y_p (Y_s) is restricted to prefixes (suffixes) of Y.

Non-flat variables occurring only in $\mathcal{H}$

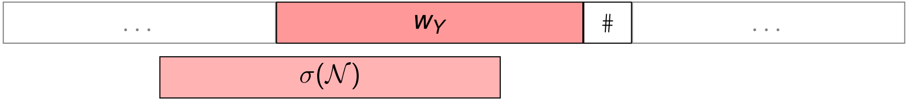
It is problematic to know why σ fails to be a model.

It would be much easier to solve modified formulae

$$\varphi_{Pref} \triangleq \neg contains(\mathcal{N}[Y/Y_p\#], \mathcal{H}[Y/Y_p\#]),$$
$$\varphi_{Suf} \triangleq \neg contains(\mathcal{N}[Y/\#Y_s], \mathcal{H}[Y/\#Y_s]).$$

where $\#$ is a fresh separator symbol and Y_p (Y_s) is restricted to prefixes (suffixes) of Y.

Intuitively, if $\sigma \not\models \varphi_{Pref}$ then we have the following situation:



Modularizing the proof

We introduce Γ_Y—a tool that allows us to solve φ_{Pref} and φ_{Suf} separately[3] and then glue together the prefix and suffix to produce $\sigma \models \neg contains(\mathcal{N}, \mathcal{H})$.

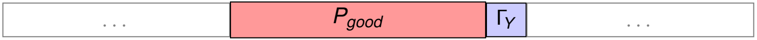
- Γ_Y is an infix that acts as a fresh separator symbol $\#$

[3] With some technical assumptions on σ'

Modularizing the proof

We introduce Γ_Y—a tool that allows us to solve φ_{Pref} and φ_{Suf} separately[3] and then glue together the prefix and suffix to produce $\sigma \models \neg contains(\mathcal{N}, \mathcal{H})$.

- Γ_Y is an infix that acts as a fresh separator symbol $\#$

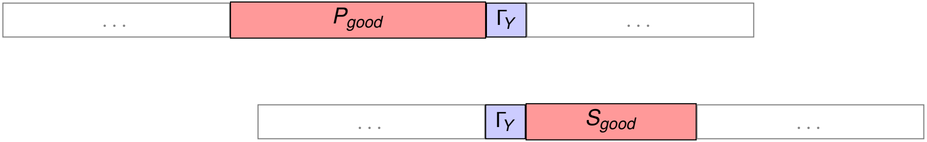


[3] With some technical assumptions on σ'

Modularizing the proof

We introduce Γ_Y—a tool that allows us to solve φ_{Pref} and φ_{Suf} separately[3] and then glue together the prefix and suffix to produce $\sigma \models \neg contains(\mathcal{N}, \mathcal{H})$.

- Γ_Y is an infix that acts as a fresh separator symbol $\#$

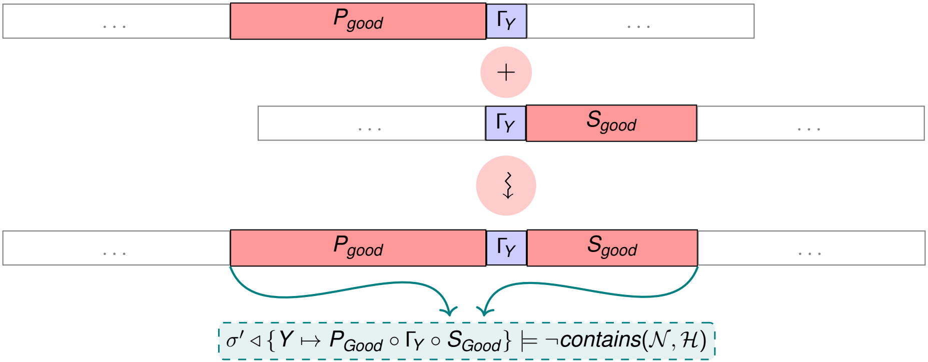


[3] With some technical assumptions on σ'

Modularizing the proof

We introduce Γ_Y—a tool that allows us to solve φ_{Pref} and φ_{Suf} separately[3] and then glue together the prefix and suffix to produce $\sigma \models \neg contains(\mathcal{N}, \mathcal{H})$.

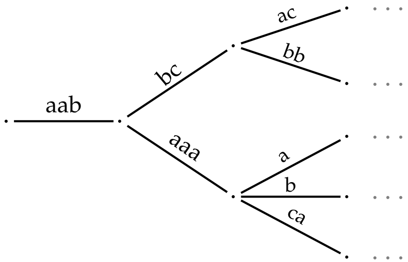
- Γ_Y is an infix that acts as a fresh separator symbol $\#$



[3] With some technical assumptions on σ'

Finding a suitable prefix

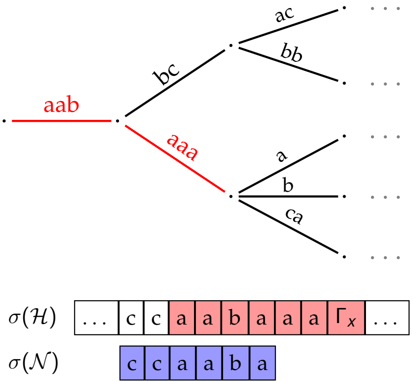
We explore prefixes of Y systematically, using a *prefix tree*.



Finding a suitable prefix

Some vertices are dead ends

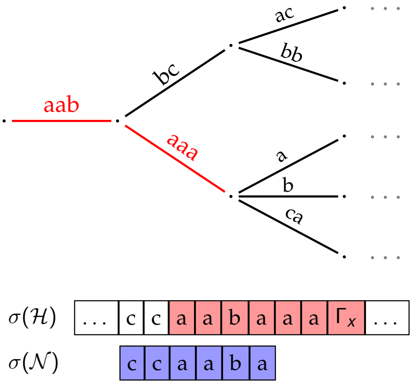
Consider the prefix aabaaa, and the following situation.



Finding a suitable prefix

Some vertices are dead ends

Consider the prefix aabaaa, and the following situation.

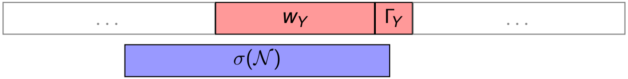


We mark some nodes as dead ends, and do not explore their successors.

Failure to find a good prefix

Special form of a solution

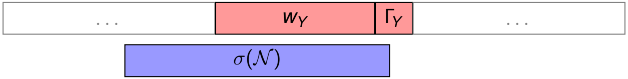
We explore prefixes in the prefix tree up to a certain bound λ.
It is useful to think of Γ_Y as a new alphabet symbol, however, it is still just a word.



Failure to find a good prefix

Special form of a solution

We explore prefixes in the prefix tree up to a certain bound λ.
It is useful to think of Γ_Y as a new alphabet symbol, however, it is still just a word.



Therefore, after exploring the prefix tree up to λ, we might be in a situation:

- We have not found a good prefix, and
- there are vertices (leading to unexplored prefixes longer than λ) that are not dead-ends.

Failure to find a good prefix

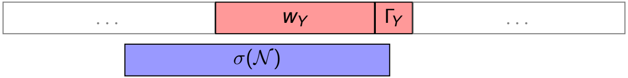
Special form of a solution

[4] Technical assumption. Justification: variables with a sorter value can be replaced by their values.

Failure to find a good prefix

Special form of a solution

Let us analyse a prefix w_Y with $|w_Y| > \lambda$ that leads to a vertex that is not marked as a dead end.

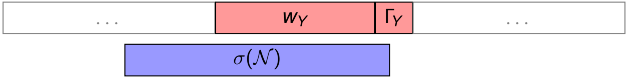


[4] Technical assumption. Justification: variables with a sorter value can be replaced by their values.

Failure to find a good prefix

Special form of a solution

Let us analyse a prefix w_Y with $|w_Y| > \lambda$ that leads to a vertex that is not marked as a dead end.



Moreover, we have the following:

1. $\mathcal{N}$ contains only flat variables, and
2. $\sigma(X)$ is longer than some constant for every flat variable X[4].

---

[4] Technical assumption. Justification: variables with a sorter value can be replaced by their values.

Failure to find a good prefix

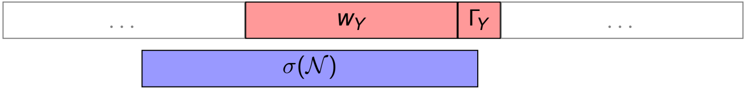
Special form of a solution (continued)

Since $\sigma \not\models \varphi_{Pref}$, we have

Failure to find a good prefix

Special form of a solution (continued)

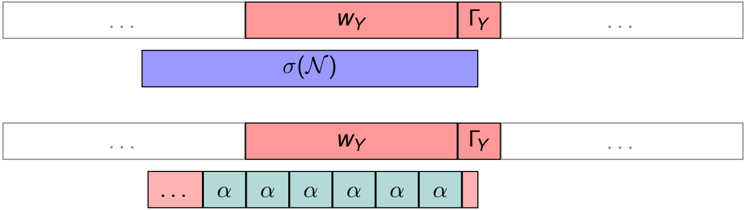
Since $\sigma \not\models \varphi_{Pref}$, we have



Failure to find a good prefix

Special form of a solution (continued)

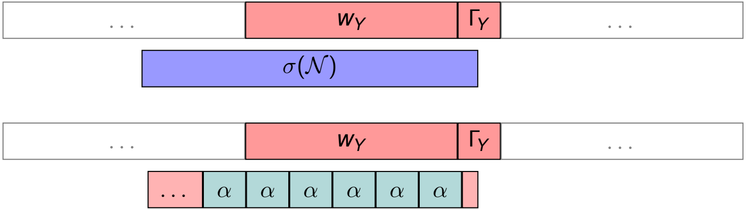
Since $\sigma \not\models \varphi_{Pref}$, we have



Failure to find a good prefix

Special form of a solution (continued)

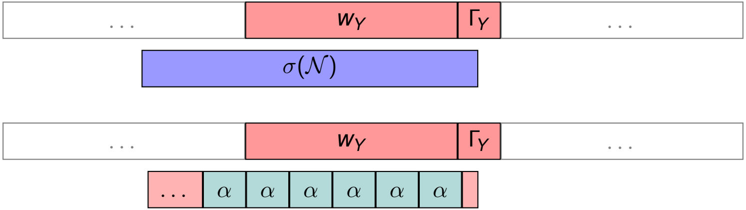
Since $\sigma \not\models \varphi_{Pref}$, we have



Failure to find a good prefix

Special form of a solution (continued)

Since $\sigma \not\models \varphi_{Pref}$, we have

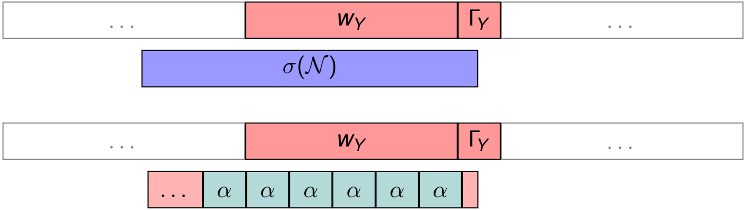


Therefore, $W_Y = s \circ \alpha^k \circ p$ for some $p \in Pref(\alpha)$, $s \in Suf(\alpha)$ and $k \in \mathbb{N}$ where $L_X = (\alpha^\ell)^*$ is the language of the rightmost variable in $\mathcal{N}$.

Failure to find a good prefix

Special form of a solution (continued)

Since $\sigma \not\models \varphi_{Pref}$, we have



Therefore, $W_Y = s \circ \alpha^k \circ p$ for some $p \in Pref(\alpha)$, $s \in Suf(\alpha)$ and $k \in \mathbb{N}$ where $L_X = (\alpha^\ell)^*$ is the language of the rightmost variable in $\mathcal{N}$.

We show that if there is a model $\sigma \triangleq \sigma' \triangleleft \{Y \mapsto w_Y\}$ with $|w_Y| > \lambda$ such that w_Y has the form $w_Y = p\alpha^k s$, then there is a model $\hat{\sigma} = \sigma' \triangleleft \{Y \mapsto w'_y\}$ with $w'_Y \in L'_Y$.

$$L'_Y \triangleq (p\alpha^* s\Gamma_Y) \cap L_Y$$

Producing a complete flat underapproximation

A complete flat underapproximation of a non-flat language L_Y is computed as

$$L'_Y \triangleq F_Y \cup \left(\mathrm{Glue}(P_Y, S_Y) \cap L_Y\right)$$

where

1. $F_Y \triangleq \{w \mid |w| \leq \lambda\}$

Producing a complete flat underapproximation

A complete flat underapproximation of a non-flat language L_Y is computed as

$$L'_Y \triangleq F_Y \cup \left(\mathrm{Glue}(P_Y, S_Y) \cap L_Y\right)$$

where

1. $F_Y \triangleq \{w \mid |w| \leq \lambda\}$
2. $P_Y \triangleq \{p \circ \Gamma_Y \mid p \in \mathit{Pref}(L_Y) \wedge |p| \leq \lambda\} \cup \bigcup_{(p,s) \in \mathit{Pref}(\alpha) \times \mathit{Suf}(\alpha)} s\alpha^* p \circ \Gamma_Y$
   - α comes from the rightmost variable in $\mathcal{N}$

Producing a complete flat underapproximation

A complete flat underapproximation of a non-flat language L_Y is computed as

$$L'_Y \triangleq F_Y \cup \left(\mathrm{Glue}(P_Y, S_Y) \cap L_Y\right)$$

where

1. $F_Y \triangleq \{w \mid |w| \leq \lambda\}$
2. $P_Y \triangleq \{p \circ \Gamma_Y \mid p \in \mathit{Pref}(L_Y) \wedge |p| \leq \lambda\} \cup \bigcup_{(p,s) \in \mathit{Pref}(\alpha) \times \mathit{Suf}(\alpha)} s\alpha^* p \circ \Gamma_Y$
   - α comes from the rightmost variable in $\mathcal{N}$
3. $S_Y \triangleq \{\Gamma_Y \circ s \mid s \in \mathit{Suf}(L_Y) \wedge |s| \leq \lambda\} \cup \bigcup_{(p,s) \in \mathit{Pref}(\beta) \times \mathit{Suf}(\beta)} \Gamma_Y \circ s\beta^* p$
   - β comes from the leftmost variable in $\mathcal{N}$

Producing a complete flat underapproximation

A complete flat underapproximation of a non-flat language L_Y is computed as

$$L'_Y \triangleq F_Y \cup \left(\mathrm{Glue}(P_Y, S_Y) \cap L_Y\right)$$

where

1. $F_Y \triangleq \{w \mid |w| \leq \lambda\}$
2. $P_Y \triangleq \{p \circ \Gamma_Y \mid p \in \mathit{Pref}(L_Y) \wedge |p| \leq \lambda\} \cup \bigcup_{(p,s) \in \mathit{Pref}(\alpha) \times \mathit{Suf}(\alpha)} s\alpha^* p \circ \Gamma_Y$
   - α comes from the rightmost variable in $\mathcal{N}$
3. $S_Y \triangleq \{\Gamma_Y \circ s \mid s \in \mathit{Suf}(L_Y) \wedge |s| \leq \lambda\} \cup \bigcup_{(p,s) \in \mathit{Pref}(\beta) \times \mathit{Suf}(\beta)} \Gamma_Y \circ s\beta^* p$
   - β comes from the leftmost variable in $\mathcal{N}$
4. $\mathrm{Glue}(p \circ \Gamma_Y, \Gamma_Y \circ s) \triangleq p \circ \Gamma_Y \circ s$

Applying underapproximation

STEP 3. UNDERAPPROXIMATE NON-FLAT VARIABLES OCCURRING ONLY $\mathcal{N}$

Input: A formula

$$\varphi \triangleq \neg contains(\mathcal{N}, \mathcal{H}) \land \bigwedge_{X \in \mathbb{X}} X \in L_X$$

with $\mathcal{N}$ containing only flat variables and a set $\mathbb{X}_{\mathcal{N}}$ of non-flat variables occurring in $\mathcal{N}$.

Output: An equisatisfiable formula

$$\varphi \triangleq \neg contains(\mathcal{N}, \mathcal{H}) \land \bigwedge_{X \in \mathbb{X} \setminus \mathbb{X}_{\mathcal{N}}} X \in L_X \land \bigwedge_{Y \in \mathbb{X}_{\mathcal{N}}} Y \in L'_Y$$

such that the language L'_Y is flat for every $Y \in \mathbb{X}_{\mathcal{N}}$.

Entire decision procedure (sketch)

1. Normalize φ_0 into a disjunction $\bigvee_{i \in I} \varphi_i$, pick a disjunct $\varphi_i = \neg \mathit{contains}(\mathcal{N}_i, \mathcal{H}_i) \land \ldots$.

Entire decision procedure (sketch)

1. Normalize φ_0 into a disjunction $\bigvee_{i \in I} \varphi_i$, pick a disjunct $\varphi_i = \neg \mathit{contains}(\mathcal{N}_i, \mathcal{H}_i) \land \ldots$.
2. If φ_i is easy, then return the solution.

Entire decision procedure (sketch)

1. Normalize φ_0 into a disjunction $\bigvee_{i \in I} \varphi_i$, pick a disjunct $\varphi_i = \neg \mathit{contains}(\mathcal{N}_i, \mathcal{H}_i) \land \ldots$.
2. If φ_i is easy, then return the solution.
3. Replace all non-flat variables occurring in both $\mathcal{N}_i$ and $\mathcal{H}_i$ by fresh alphabet symbols.

Entire decision procedure (sketch)

1. Normalize φ_0 into a disjunction $\bigvee_{i \in I} \varphi_i$, pick a disjunct $\varphi_i = \neg \mathit{contains}(\mathcal{N}_i, \mathcal{H}_i) \land \ldots$.
2. If φ_i is easy, then return the solution.
3. Replace all non-flat variables occurring in both $\mathcal{N}_i$ and $\mathcal{H}_i$ by fresh alphabet symbols.
4. Replace languages of remaining non-flat variables occurring in $\mathcal{H}$ with their flat underapproximations.

Entire decision procedure (sketch)

1. Normalize φ_0 into a disjunction $\bigvee_{i \in I} \varphi_i$, pick a disjunct $\varphi_i = \neg \mathit{contains}(\mathcal{N}_i, \mathcal{H}_i) \land \ldots$.
2. If φ_i is easy, then return the solution.
3. Replace all non-flat variables occurring in both $\mathcal{N}_i$ and $\mathcal{H}_i$ by fresh alphabet symbols.
4. Replace languages of remaining non-flat variables occurring in $\mathcal{H}$ with their flat underapproximations.
5. Solve resulting formula by reduction to Presburger arithmetic.

Entire decision procedure (sketch)

1. Normalize φ_0 into a disjunction $\bigvee_{i \in I} \varphi_i$, pick a disjunct $\varphi_i = \neg \mathit{contains}(\mathcal{N}_i, \mathcal{H}_i) \land \dots$.
2. If φ_i is easy, then return the solution.
3. Replace all non-flat variables occurring in both $\mathcal{N}_i$ and $\mathcal{H}_i$ by fresh alphabet symbols.
4. Replace languages of remaining non-flat variables occurring in $\mathcal{H}$ with their flat underapproximations.
5. Solve resulting formula by reduction to Presburger arithmetic.

Resulting complexity is EXPSPACE.

Entire decision procedure (sketch)

1. Normalize φ_0 into a disjunction $\bigvee_{i \in I} \varphi_i$, pick a disjunct $\varphi_i = \neg\mathit{contains}(\mathcal{N}_i, \mathcal{H}_i) \wedge \ldots$.
2. If φ_i is easy, then return the solution.
3. Replace all non-flat variables occurring in both $\mathcal{N}_i$ and $\mathcal{H}_i$ by fresh alphabet symbols.
4. Replace languages of remaining non-flat variables occurring in $\mathcal{H}$ with their flat underapproximations.
5. Solve resulting formula by reduction to Presburger arithmetic.

Resulting complexity is EXPSPACE.

Future work:

- Extend our proof to cover conjunctions of $\neg$*contains*.
- Improve the complexity bounds of our result.

Conclusion: Although $\neg$*contains* hides quantifiers inside, it is *decidable*.

Conclusion: Although $\neg$*contains* hides quantifiers inside, it is *decidable*.

Thank you for you attention.
Questions?