Validating Library Usage Interactively

William Harris, Guoliang Jin, Shan Lu, and Somesh Jha

THE UNIVERSITY of WISCONSIN MADISON

Outline

1. **Motivation**
2. Problem definition
3. Technique and experiments

Outline

1. Motivation
2. Problem definition
3. Technique and experiments

Outline

1. Motivation

3. Technique and experiments

Outline

1. Motivation

Developers Often Optimize Library Usage

Developers Often Optimize Library Usage

In a previous study (*Understanding and Detecting Real-World Performance Bugs*, Jin et. al., PLDI '12), of 109 bug patches:

Developers Often Optimize Library Usage

In a previous study (*Understanding and Detecting Real-World Performance Bugs*, Jin et. al., PLDI '12), of 109 bug patches:

42

caused by

"an inefficient function call sequence composed of efficient individual functions"

Developers Often Optimize Library Usage

In a previous study (*Understanding and Detecting Real-World Performance Bugs*, Jin et. al., PLDI '12), of 109 bug patches:

34

caused by

"calling functions that conduct unnecessary work given the calling context"

Developers Often Optimize Library Usage *Concisely*

In a previous study (*Understanding and Detecting Real-World Performance Bugs*, Jin et. al., PLDI '12), of 109 bug patches:

42

contained ≤ 5 lines of code,
and median size was 8 lines

The Case for Verification

Opportunities to improve practice: optimizations...

- are discussed extensively
- validated by testing only

Why its feasible: optimizations...

- use a clean interface
- are small

The Case for Verification

Opportunities to improve practice: optimizations...

- are discussed extensively
- validated by testing only

Why its feasible: optimizations...

- use a clean interface
- are small

Why its hard: optimizations implicitly rely on subtle library properties

Mozilla Bug #409961

Original

```
DbList dbnewList;
Card card;
for (pos=1;pos<=numAddrs;pos++)
  card = CreateCard(user(pos));
  dbnewList = DbList();
  AddAddrToList(dbnewList,card);
return card;
```

Optimized

Mozilla Bug #409961

Original

```
DbList dbnewList;
Card card;
for (pos=1;pos<=numAddrs;pos++)
  card = CreateCard(user(pos));
  dbnewList = DbList();
  AddAddrToList(dbnewList,card);
return card;
```

Optimized

Mozilla Bug #409961

$\forall$ x, y, z.

`AddAddressToList[arg2]`(x, y)

= `AddAddressToList[arg2]`(z, y)

Original

```
DbList dbnewList;
Card card;
for (pos=1;pos<=numAddrs;pos++)
  card = CreateCard(user(pos));
  dbnewList = DbList();
  AddAddrToList(dbnewList,card);
return card;
```

Optimized

MySQL Bug #38769

Original

```
for (n=0; n < 256; ++n) {
  Snapshot_info* i = at(info, n);
  if (!i) continue;
  process_snapshot(i);
}
```

Optimized

MySQL Bug #38769

Original

```
for (n=0; n<< 256; ++n) {    ;++n){
  Snapshot_info* i = at(info, n);
  if (!i) continue;
  process_snapshot(i);
}
```

Optimized

MySQL Bug #38769

Original

```
for (n=0; n<< 256; ++n) {    ;++n){
  Snapshot_info* i = at(info, n);
  if (!i) continue;
  process_snapshot(i);
}
```

Optimized

```
snap_count(info)
```

MySQL Bug #38769

$$\forall\, x.\ \texttt{snap\_count}(x) < 256$$

$$\forall\, x, y.\ \texttt{snap\_count}(x) \leq y \Rightarrow \texttt{at}(x, y) = \text{NULL}$$

Original

```
for (n=0; n<< 256; ++n) {    ;++n){
  Snapshot_info* i = at(info, n);
  if (!i) continue;
  process_snapshot(i);
}
```

Optimized

```
snap_count(info)
```

Apache Bug #34464

Original

```
string sb := “”;
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

Optimized

Apache Bug #34464

Original

```
string sb := “”;
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

Optimized

```
int pos := -len(s);
       pos < 0
       ||          sub(   pos),

   pos := len(sb) - len(s);
```

Apache Bug #34464

$$\texttt{len}(\text{“”}) = 0$$

$$\forall\, x, y.\ \texttt{len}(x) > \texttt{len}(y) \Rightarrow \neg\texttt{is\_sub}(x, y)$$

$$\forall\, x, y, z.\ \neg\texttt{is\_sub}(x, y) \Rightarrow$$

$$\texttt{is\_sub}(\texttt{app}(x, z), y) \Leftrightarrow$$

$$\texttt{is\_sub}(\texttt{sub}(\texttt{app}(x, z), \texttt{len}(\texttt{app}(x, z)) - \texttt{len}(y)), y)$$

Original

```
string sb := “”;
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

Optimized

```
int pos := -len(s);
      pos < 0
      ||        sub(  pos),

pos := len(sb) - len(s);
```

Outline

1. Motivation

2. Problem definition
1. Problem definition
3. Technique and experiments

Outline

2. Problem definition
1. Problem definition
3. Technique and experiments

Outline

1. Problem definition
3. Technique and experiments

Outline

1. Problem definition

Apache Bug #34464

Org.

```
string sb := “”;
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

Opt.

```
string sb := “”;
int pos = -len(s);
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

Org.

```
string sb := “”;
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

Opt.

```
string sb := “”;
int pos = -len(s);
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs

Org.

```
string sb := “”;
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

Opt.

```
string sb := “”;
int pos = -len(s);
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

||

chklibs

Org.

```
string sb := "";
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

Opt.

```
string sb := "";
int pos = -len(s);
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos = len(sb) - len(s);
}
return sb;
```

User

chklibs

Org.

```
∀ x, y. is_sub(x, y)

string sb := “”;
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

Opt.

```
string sb := “”;
int pos = -len(s);
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos = strlen(sb) - len(s);
}
return sb;
```

User

chklibs

Org.

Opt.

```
∀ x, y. is_sub(x, y)
```

```
string sb := “”;
while (!is_sub(sb,
  app(sb, get());
}
return sb;
```

Invalid

```
string sb := “”;
int pos = -len(s);
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos = len(sb) - len(s);
}
return sb;
```

User

chklibs

Org.

```
∀ x, y. is_sub(x, y)

string sb := “”;
while (!is_sub(sb,
  app(sb, get());
}
return sb;
```

Invalid

Opt.

```
∀ x, y. len(x) > len(y) ⇒
  ¬is_sub(x, y)   ";
int pos = -len(s);
while (pos < 0 ||
    !is_sub(sub(sb, pos), s)){
  app(s
  pos
}
return sb;
```

b) - len(s);

User

Org. ≈ Opt
assuming
∀ x, y. `len`(x) > `len`(y) ⇒
¬`is_sub`(x, y)
...

`chklibs`

Org.

∀ x, y. `is_sub`(x, y)

```
string sb := "";
while (!is_sub(sb,
  app(sb, get());
}
return sb;
```

Invalid

Opt.

∀ x, y. `len`(x) > `len`(y) ⇒
¬`is_sub`(x, y)

```
";
int pos = -len(
while (pos < 0
    !is_sub(sub(sb, pos), s)){
  app(s
  pos
}
return sb;
```

...

Valid

User

`chklibs`
Requirements

`chklibs` Requirements

1. Relative soundness

`chklibs` Requirements

1. Relative soundness

`chklibs` Requirements

1. Relative soundness

1. Non-redundant queries

2. Queries on properties of the library only

`chklibs` Requirements

1. Non-redundant queries

1. Non-redundant queries

Inputs

```
string sb := "";
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := "";
int pos = -len(s);
while (pos < 0
       !is_sub(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs 

User 

1. Non-redundant queries

Inputs

```
string sb := "";
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := "";
int pos = -len(s);
while (pos < 0
       !is_sub(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs 

∀ x, y. `len`(x) > `len`(y) ⇒

`is_sub`(x, y)

User 

1. Non-redundant queries

Inputs

```
string sb := "";
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := "";
int pos = -len(s);
while (pos < 0
       !is_sub(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs

∀ x, y. `len`(x) > `len`(y) ⇒

`is_sub`(x, y)

Invalid

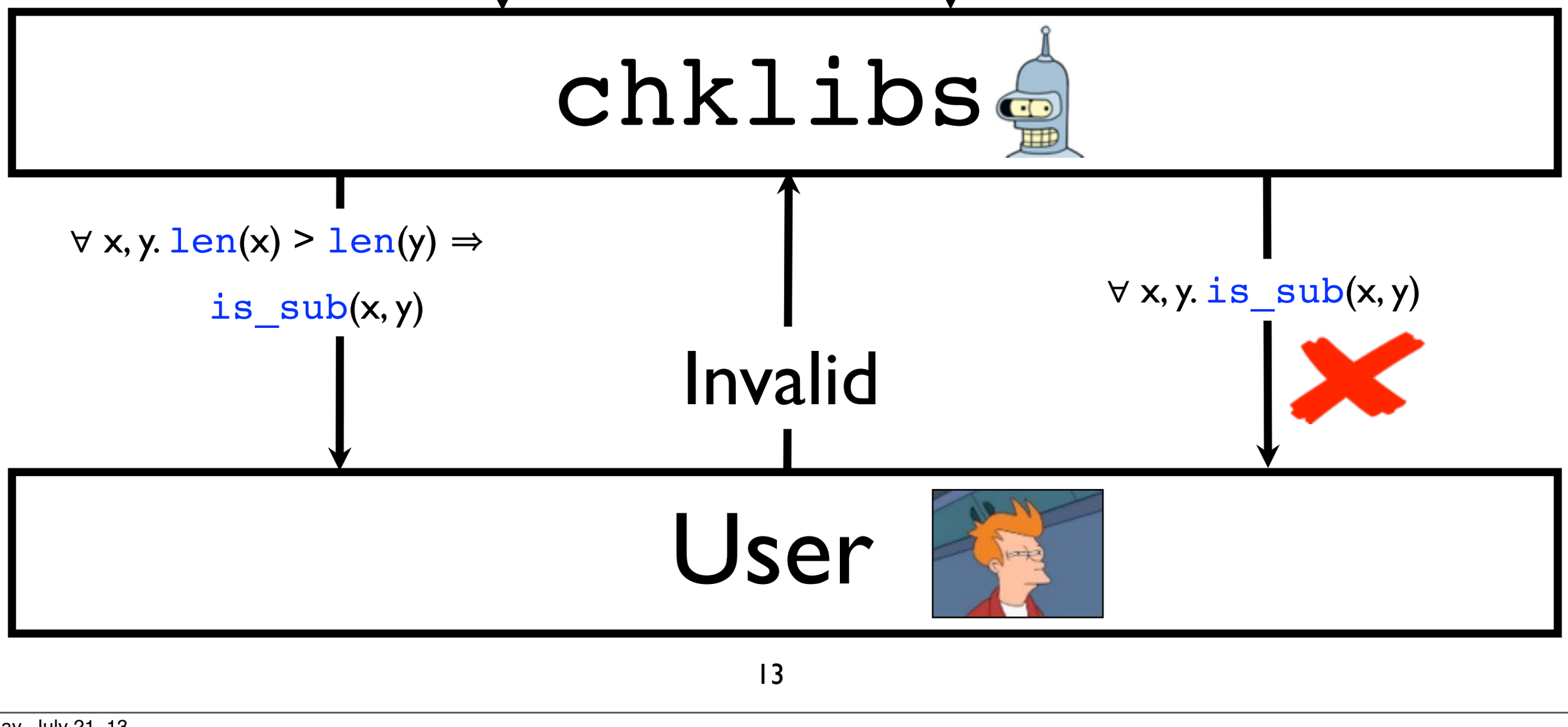
User

1. Non-redundant queries

Inputs

```
string sb := "";
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := "";
int pos = -len(s);
while (pos < 0
       !is_sub(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```



1. Queries on properties of the library only

Inputs

```
string sb := "";
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := "";
int pos = -len(s);
while (pos < 0
       !is_sub(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs 

User 

1. Queries on properties of the library only

Inputs

```
string sb := "";
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := "";
int pos = -len(s);
while (pos < 0
       !is_sub(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs 

$\forall$ x, y. `len`(x) > `len`(y) $\Rightarrow$

$\neg$`is_sub`(x, y)

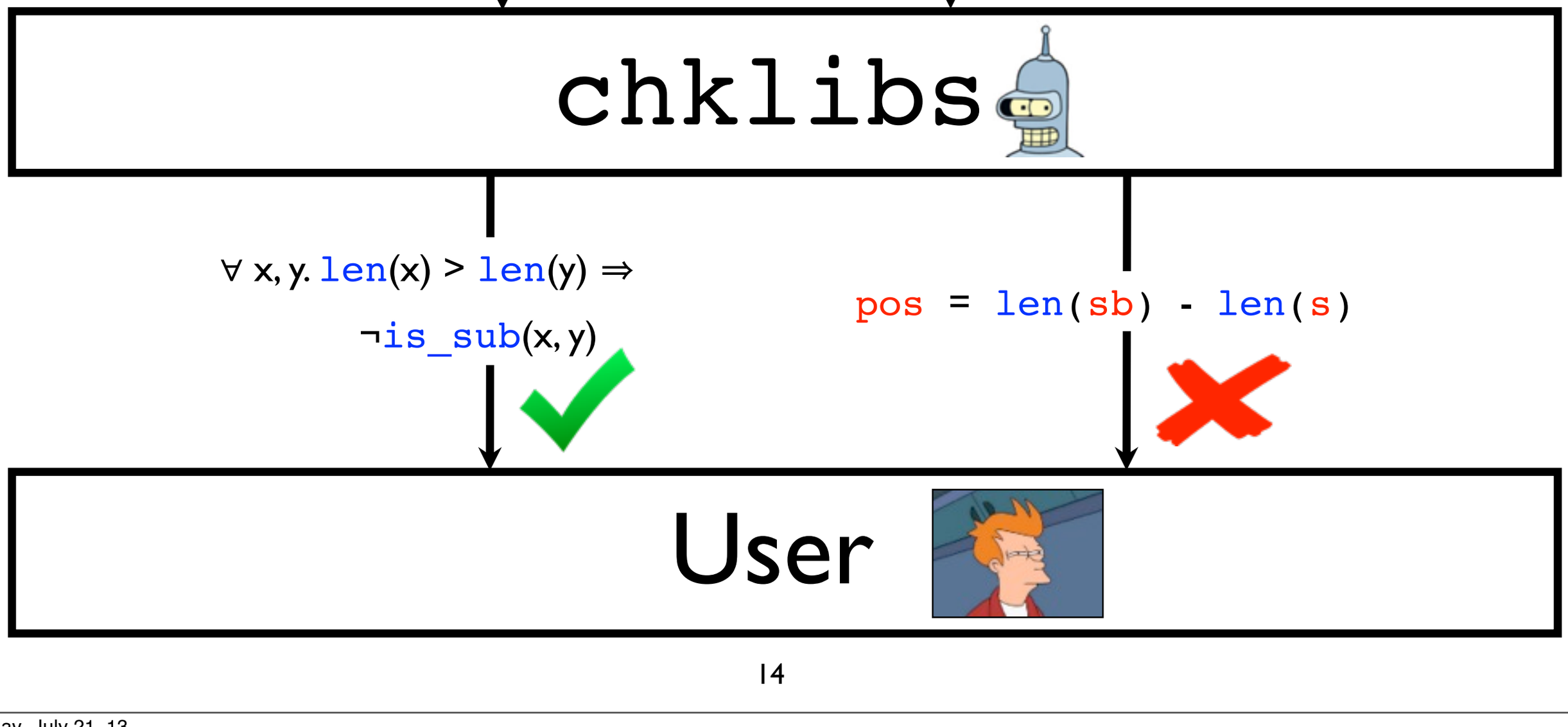
User 

1. Queries on properties of the library only

Inputs

```
string sb := "";
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := "";
int pos = -len(s);
while (pos < 0
       !is_sub(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```



Why involve the user?

- Libraries may be difficult to analyze
- Do small optimizations contain specifications of big callees?

Outline

1. Motivation
2. Problem definition
3. **Technique** and experiments

Outline

2. Problem definition

3. **Technique** and experiments

Outline

B. **Technique** and experiments

Outline

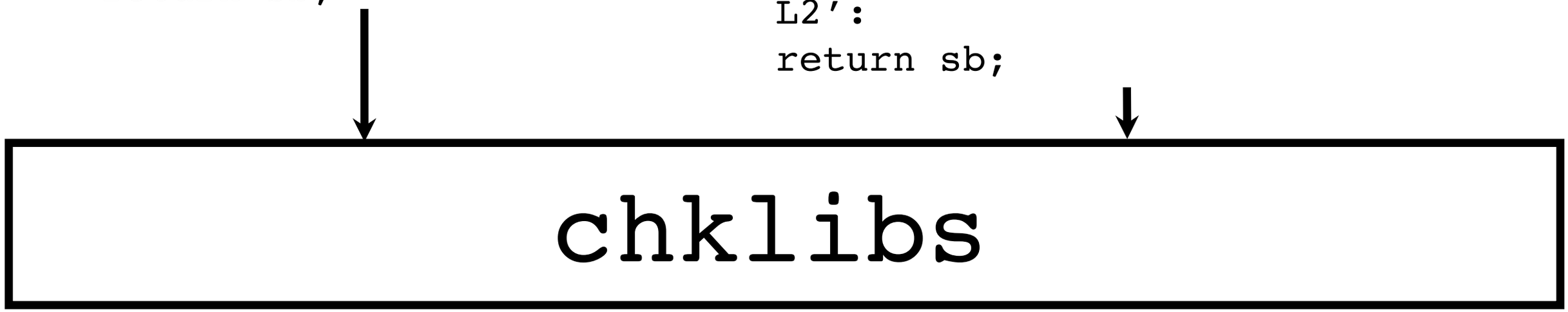
1. Technique and experiments

Technique

- Based on known translation-validation algorithm (Necula, PLDI '00)
- Key difference: simultaneously find formula in simulation relation and supporting library property over predicates in post

```
L0:
string sb := “”;
L1:
while (!is_sub(sb, s)) {
  app(sb, get());
}
L2:
return sb;
```

```
L0’:
string sb := “”;
int pos = -len(s);
L1’:
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
L2’:
return sb;
```



User

chklibs

```
L0:
string sb := “”;
L1:
while (!is_sub(sb, s)) {
  app(sb, get());
}
L2:
return sb;
```

```
L0’:
string sb := “”;
int pos = -len(s);
L1’:
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
L2’:
return sb;
```

validated Lib. property: True

User

chklibs

```
L0:
string sb := “”;
L1:
while (!is_sub(sb, s)) {
  app(sb, get());
}
L2:
return sb;
```

```
L0’:
string sb := “”;
int pos = -len(s);
L1’:
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
L2’:
return sb;
```

validated Lib. property: True

User

chklibs

```
                                  s = s'
                                  L0':
L0:                               string sb := "";
string sb := "";                  int pos = -len(s);
L1:                               L1':
while (!is_sub(sb, s)) {          while (pos < 0 ||
  app(sb, get());                        !is_sub(sub(sb, pos), s)){
}                                   app(sb, get());
L2:                                 pos := len(sb) - len(s);
return sb;                        }
                                  L2':
                                   return sb;
```

validated Lib. property: True

User

chklibs

s = s'

```
L0:
string sb := "";
L1:
while (!is_sub(sb, s)) {
  app(sb, get());
}
L2:
return sb;
```

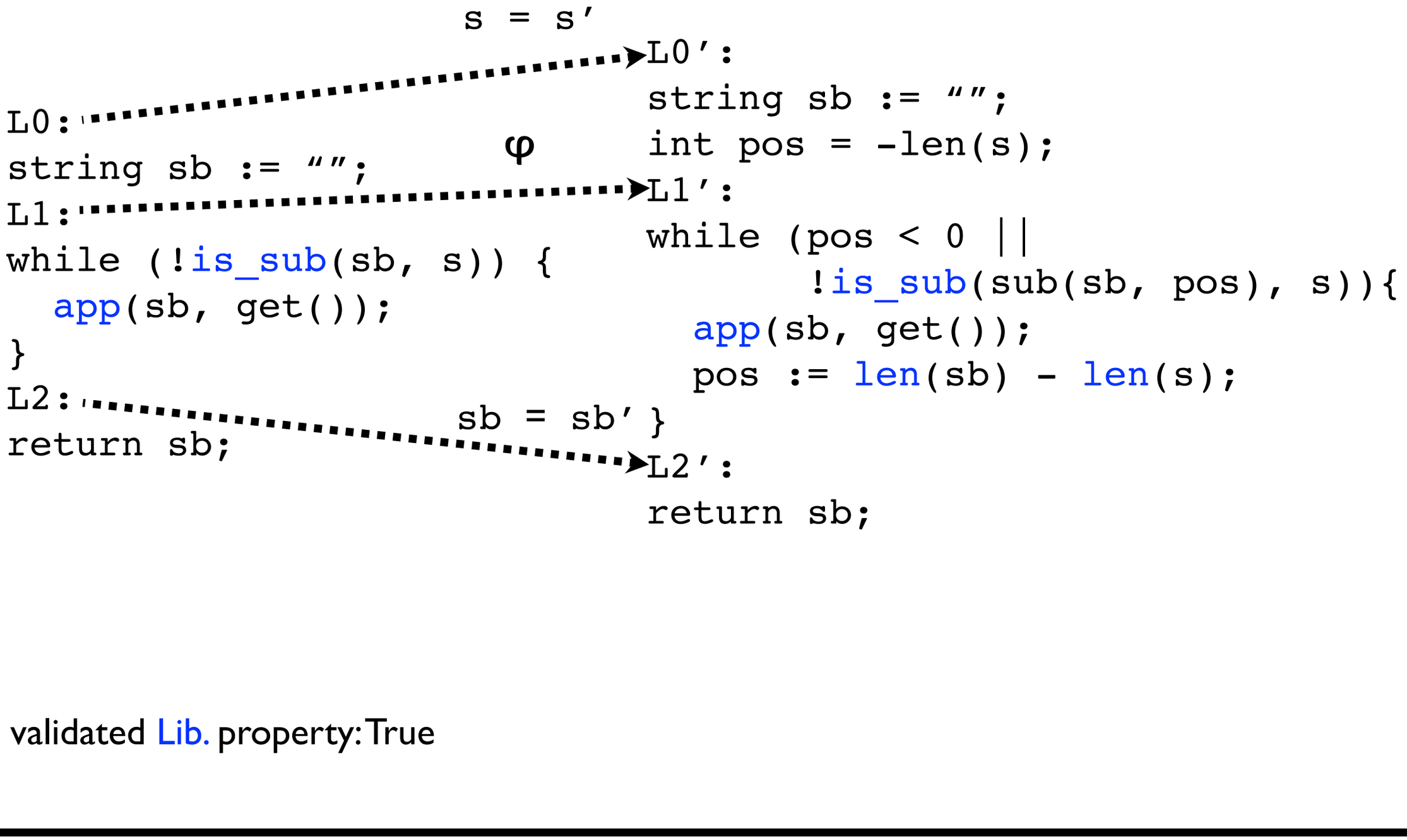
sb = sb'

```
L0':
string sb := "";
int pos = -len(s);
L1':
while (pos < 0 ||
         !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
L2':
return sb;
```

validated Lib. property: True

User

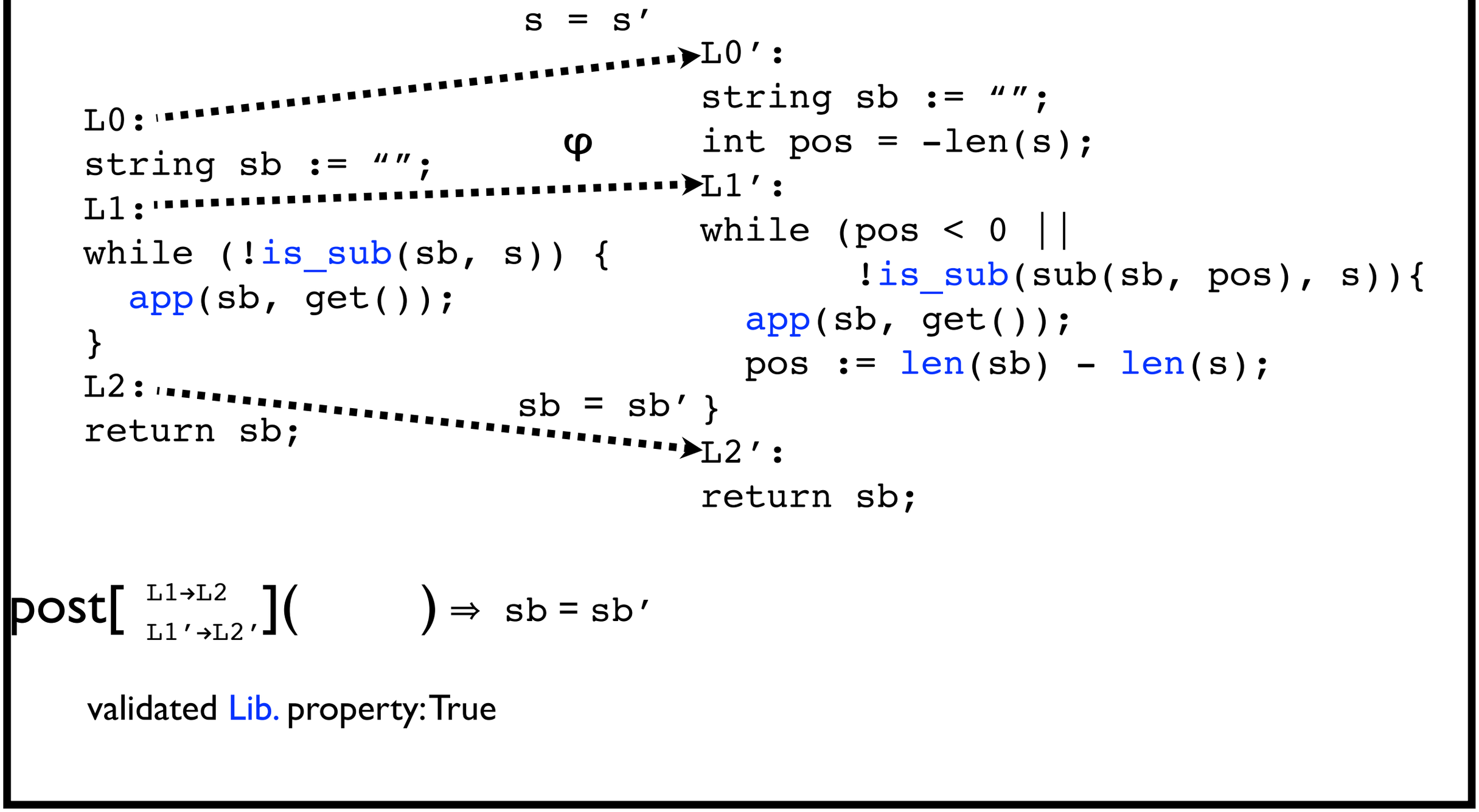
chklibs



validated Lib. property: True

User

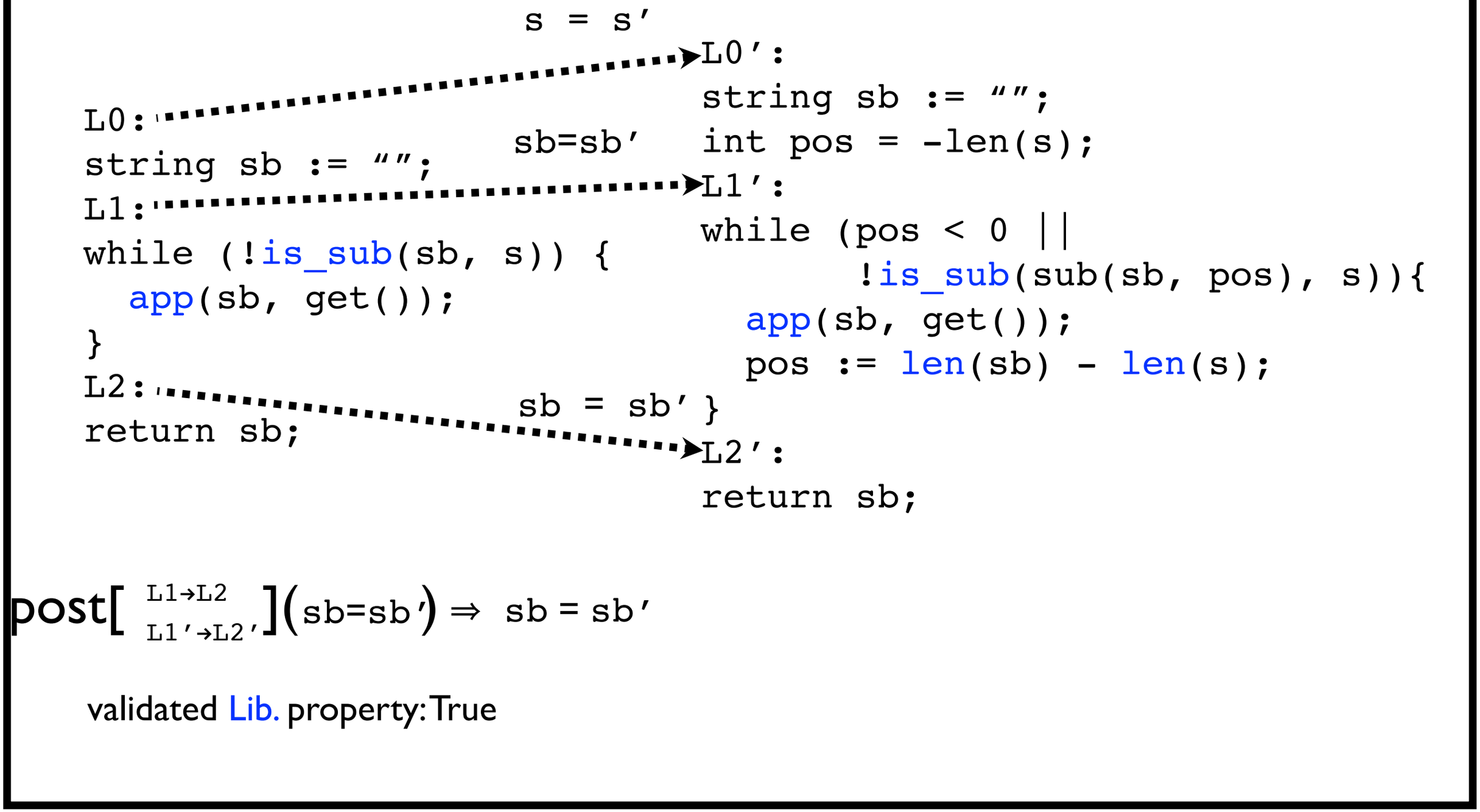
chklibs



validated Lib. property: True

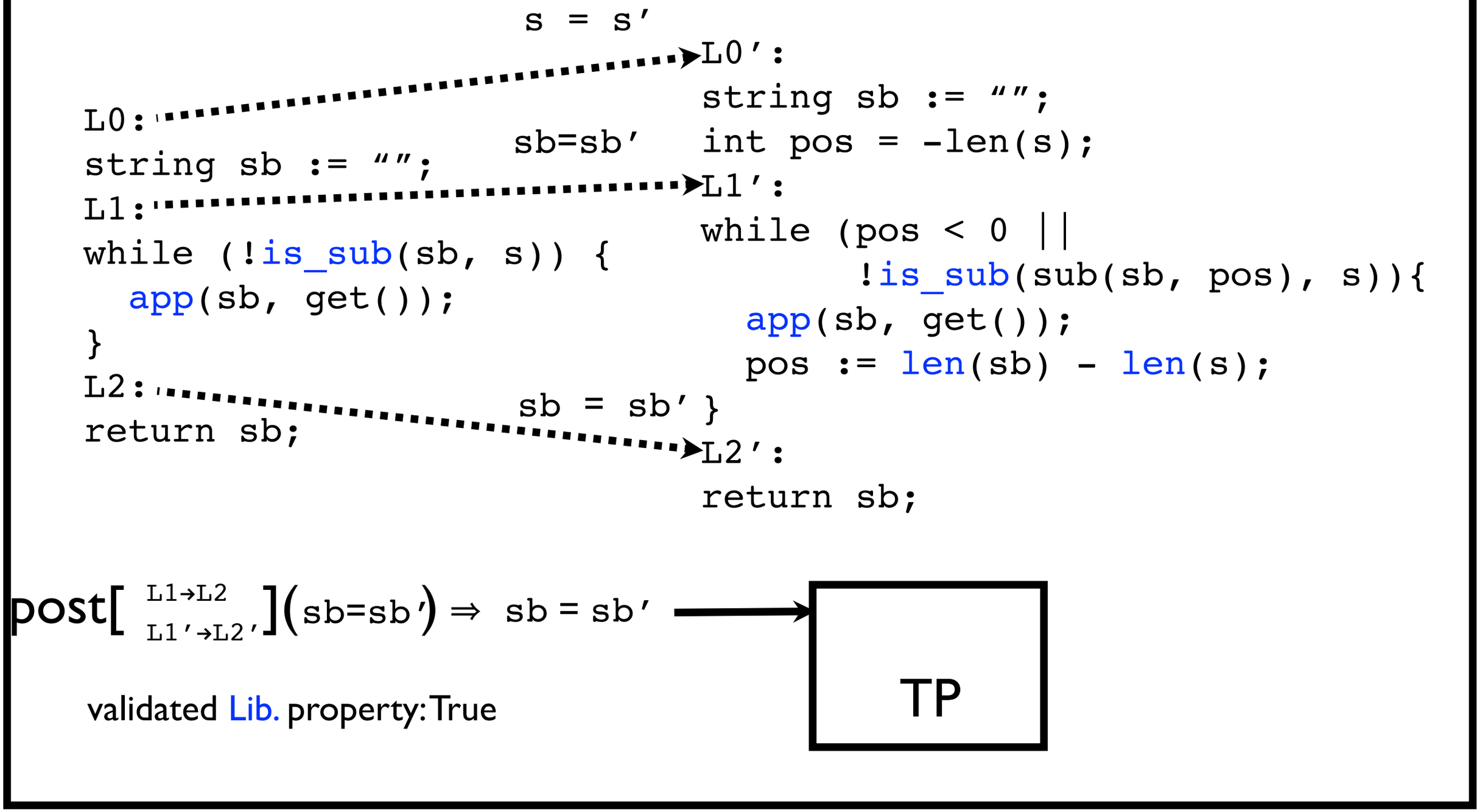
User

chklibs



User

chklibs



User

chklibs

s = s′

sb=sb′

sb = sb′

```
L0:
string sb := "";
L1:
while (!is_sub(sb, s)) {
  app(sb, get());
}
L2:
return sb;
```

```
L0':
string sb := "";
int pos = -len(s);
L1':
while (pos < 0 ||
          !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
L2':
return sb;
```

$\text{post}[\,{}^{L1 \to L2}_{L1' \to L2'}\,](sb = sb') \Rightarrow sb = sb'$ → TP → Invalid

validated Lib. property: True

User

chklibs

```
L0:
string sb := “”;
L1:
while (!is_sub(sb, s)) {
  app(sb, get());
}
L2:
return sb;
```

s = s′ (L0 → L0′)

sb=sb′ (L1 → L1′)

sb = sb′ (L2 → L2′)

```
L0′:
string sb := “”;
int pos = -len(s);
L1′:
while (pos < 0 ||
        !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
L2′:
return sb;
```

$\text{post}\left[\begin{smallmatrix}\texttt{L1}\to\texttt{L2} \\ \texttt{L1'}\to\texttt{L2'}\end{smallmatrix}\right](\texttt{sb=sb'}) \Rightarrow \texttt{sb = sb'}$ → TP

validated Lib. property: True

User

chklibs

s = s′

sb=sb′

sb = sb′

```
L0:
string sb := "";
L1:
while (!is_sub(sb, s)) {
  app(sb, get());
}
L2:
return sb;
```

```
L0':
string sb := "";
int pos = -len(s);
L1':
while (pos < 0 ||
          !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
L2':
return sb;
```

$\mathrm{post}\left[\begin{smallmatrix} L1 \to L2 \\ L1' \to L2' \end{smallmatrix}\right](sb = sb') \Rightarrow sb = sb'$ → Lib.-Abd. TP

validated Lib. property: True → Lib.-Abd. TP

User

chklibs

```
L0:                          s = s'        L0':
string sb := "";                           string sb := "";
                             sb=sb'        int pos = -len(s);
L1:                                        L1':
while (!is_sub(sb, s)) {                   while (pos < 0 ||
  app(sb, get());                                !is_sub(sub(sb, pos), s)){
}                                            app(sb, get());
                                             pos := len(sb) - len(s);
L2:                          sb = sb'      }
return sb;                                 L2':
                                           return sb;
```

post[L1→L2 / L1'→L2'](sb=sb') ⇒ sb = sb' → **Lib.-Abd. TP** → ∀x, y. find(x, y)

validated Lib. property: True → **Lib.-Abd. TP**

User

chklibs

```
L0:
string sb := “”;
L1:
while (!is_sub(sb, s)) {
  app(sb, get());
}
L2:
return sb;
```

s = s′ · sb=sb′ · sb = sb′

```
L0′:
string sb := “”;
int pos = -len(s);
L1′:
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
L2′:
return sb;
```

post[L1→L2 / L1′→L2′](sb=sb′) ⇒ sb = sb′

validated Lib. property: True

Lib.-Abd. TP

∀x, y. find(x, y)

User

Experimental Highlights

- Ran tool on 11 performance-bug patches from Apache, Mozilla, and MySQL
- On average, found supporting conditions
  - in < 1 second
  - with < 10 queries (0 - 7 invalid)

Thanks for your attention!

Questions?

```
string sb := “”;
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := “”;
int pos = -len(s);
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs 

User 

Questions?

```
string sb := “”;
while (!is_sub(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := “”;
int pos = -len(s);
while (pos < 0 ||
       !is_sub(sub(sb, pos), s)){
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

Org. ≈ Opt
assuming
∀ x, y. `len`(x) > `len`(y) ⇒
¬`is_sub`(x, y)
…

`chklibs`

∀ x, y. `is_sub`(x, y)

Invalid

∀ x, y. `len`(x) > `len`(y) ⇒
¬`is_sub`(x, y)

…

Valid

User

Extra Slides

Apache Bug #19101

Original

```
vector<BuildListener> ls =
getBuildListeners();
for (i=0; i < size(ls); i++) {
  BuildListener l = at(ls, i);
  l.startTask();
}
```

Optimized

Apache Bug #19101

Original

```
vector<BuildListener> ls =
getBuildListeners();
for (i=0; i < size(ls); i++) {
  BuildListener l = at(ls, i);
  l.startTask();
}
```

Optimized

Apache Bug #19101

Original

```
vector<BuildListener> ls =
getBuildListeners();
for (i=0; i < size(ls); i++) {
  BuildListener l = at(ls, i);
  l.startTask();
}
```

Optimized

```
int sz =          ;
sz
```

Apache Bug #19101

$\forall$ x, y. `size`(`at[arg0]`(x, y)) = `size`(x)

Original

```
vector<BuildListener> ls =
getBuildListeners();
for (i=0; i < size(ls); i++) {
  BuildListener l = at(ls, i);
  l.startTask();
}
```

Optimized

```
int sz =          ;
sz
```

1. Non-redundant

```
string sb := “”;
while (!find(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := “”;
int pos = -len(s);
while (pos < 0
       !find(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs

User

1. Non-redundant

```
string sb := “”;
while (!find(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := “”;
int pos = -len(s);
while (pos < 0
       !find(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs

∀ x, y. len(x) > len(y) ⇒

¬find(x, y)

User

1. Non-redundant

```
string sb := “”;
while (!find(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := “”;
int pos = -len(s);
while (pos < 0
       !find(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs

∀ x, y. len(x) > len(y) ⇒

¬find(x, y)

Valid

User

1. Non-redundant

```
string sb := “”;
while (!find(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := “”;
int pos = -len(s);
while (pos < 0
       !find(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs

∀ x, y. len(x) > len(y) ⇒

¬find(x, y)

∀ x, y, z. ¬find(app(x, y), z) ⇒

¬ find(x, z)

Valid

User

1. Non-redundant

```
string sb := “”;
while (!find(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := “”;
int pos = -len(s);
while (pos < 0
       !find(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs

∀ x, y. len(x) > len(y) ⇒

¬find(x, y)

Valid

∀ x, y, z. ¬find(app(x, y), z) ⇒

¬ find(x, z)

Valid

User

1. Non-redundant

```
string sb := “”;
while (!find(sb, s)) {
  app(sb, get());
}
return sb;
```

```
string sb := “”;
int pos = -len(s);
while (pos < 0
       !find(sub(sb, pos), s)) {
  app(sb, get());
  pos := len(sb) - len(s);
}
return sb;
```

chklibs

∀ x, y. len(x) > len(y) ⇒ ¬find(x, y)

Valid

∀ x, y, z. ¬find(app(x, y), z) ⇒ ¬ find(x, z)

∀ x, y, z. len(app(x, y)) > len(z) ⇒ ¬ find(x, z)

Valid

User