

Automatically Proving Memory Safety and Termination of C-Programs

Alexander Weinert
RWTH Aachen University

Research Area Computer Science 2
Published: (Termination with Pointer Arithmetic, Ströder et al., 2014)

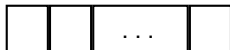
Feb 23, 2015

An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(++s));  
    return s-str;  
}
```

An Example

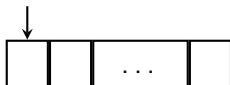
```
int strlen(char* str) {  
    char* s = str;  
    while(*(++s));  
    return s-str;  
}
```



An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(++s));  
    return s-str;  
}
```

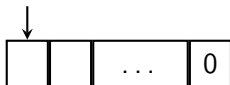
str



An Example

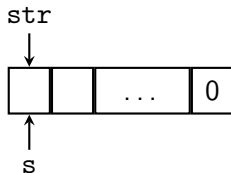
```
int strlen(char* str) {  
    char* s = str;  
    while(*(++s));  
    return s-str;  
}
```

str



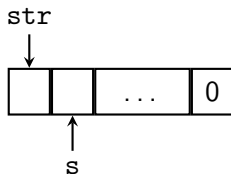
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(++s));  
    return s-str;  
}
```



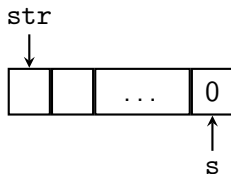
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(++s));  
    return s-str;  
}
```



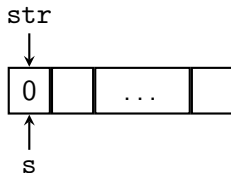
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(++s));  
    return s-str;  
}
```



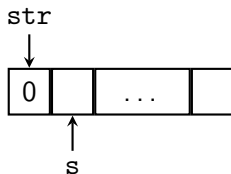
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(++s));  
    return s-str;  
}
```



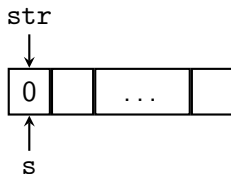
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(++s));  
    return s-str;  
}
```



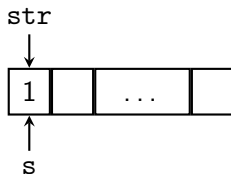
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while((*s)++);  
    return s-str;  
}
```



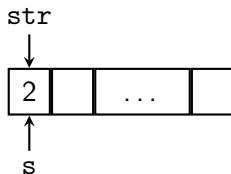
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while((*s)++);  
    return s-str;  
}
```



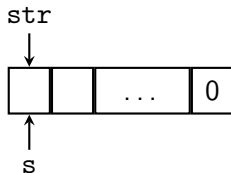
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while((*s)++);  
    return s-str;  
}
```



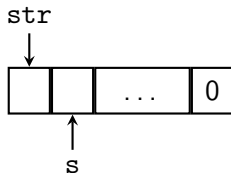
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(s++));  
    return s-str;  
}
```



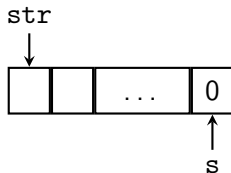
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(s++));  
    return s-str;  
}
```



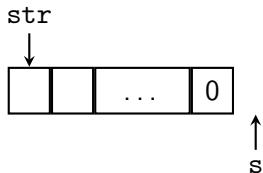
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(s++));  
    return s-str;  
}
```



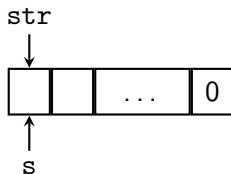
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*(s++));  
    return s-str;  
}
```



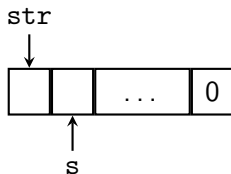
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*s) s++;  
    return s-str;  
}
```



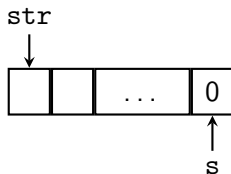
An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*s) s++;  
    return s-str;  
}
```



An Example

```
int strlen(char* str) {  
    char* s = str;  
    while(*s) s++;  
    return s-str;  
}
```



Big Picture

THE
C
PROGRAMMING
LANGUAGE

YES/NO/
MAYBE

Big Picture

THE
C
PROGRAMMING
LANGUAGE



YES/NO/
MAYBE

Big Picture

THE
C
PROGRAMMING
LANGUAGE



YES/NO/
MAYBE



LLVM

Big Picture

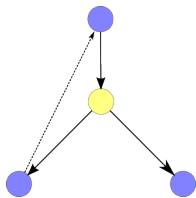
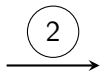
THE
C
PROGRAMMING
LANGUAGE



YES/NO/
MAYBE



LLVM



Symbolic Execution Graph

Big Picture

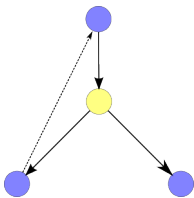
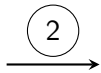
THE
C
PROGRAMMING
LANGUAGE



YES/NO/
MAYBE



LLVM



Symbolic Execution Graph



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

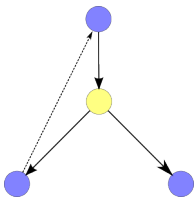
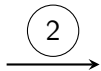
Integer Transition
System

Big Picture

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

YES/NO/
MAYBE

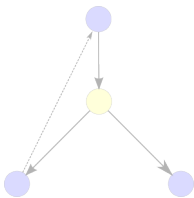


C to LLVM

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

YES/NO/
MAYBE



LLVM Compiler Infrastructure

```
int strlen(char *str) {  
    char *s = str;  
    ...  
}
```

C-program

LLVM Compiler Infrastructure

```
int strlen(char *str) {  
    char *s = str;  
    ...  
}
```

C-program

```
define i32 @strlen(i8* @str) {  
    c0 = load i8* @str  
    ...  
}
```

LLVM Internal
Representation

LLVM Compiler Infrastructure

```
int strlen(char *str) {  
    char *s = str;  
    ...  
}
```

C-program

```
define i32 @strlen(i8* @str) {  
    c0 = load i8* @str  
    ...  
}
```

LLVM Internal
Representation

```
strlen_entry:  
    push edi  
    ...  
}
```

Assembler

LLVM Compiler Infrastructure

```
int strlen(char *str) {  
    char *s = str;  
    ...  
}
```

C-program

```
define i32 @strlen(i8* @str) {  
    @c0 = load i8* @str  
    ...  
}
```

LLVM Internal
Representation

```
strlen_entry:  
    push edi  
    ...
```

Assembler

Some LLVM Instructions

- ▶ Control Flow Instructions
 - ▶ `ret`, `br`, `call`, ...

Some LLVM Instructions

- ▶ Control Flow Instructions
 - ▶ `ret`, `br`, `call`, ...
- ▶ Arithmetic Instructions
 - ▶ `add`, `icmp`, ...

Some LLVM Instructions

- ▶ Control Flow Instructions
 - ▶ ret, br, call, ...
- ▶ Arithmetic Instructions
 - ▶ add, icmp, ...
- ▶ Bitwise Instructions
 - ▶ shl, and, ...

Some LLVM Instructions

- ▶ Control Flow Instructions

- ▶ `ret`, `br`, `call`, ...

- ▶ Arithmetic Instructions

- ▶ `add`, `icmp`, ...

- ▶ Bitwise Instructions

- ▶ `shl`, `and`, ...

- ▶ Memory Instructions

- ▶ `alloca`, `load`, `store`, ...

Some LLVM Instructions

- ▶ Control Flow Instructions
 - ▶ `ret`, `br`, `call`, ...
- ▶ Arithmetic Instructions
 - ▶ `add`, `icmp`, ...
- ▶ Bitwise Instructions
 - ▶ `shl`, `and`, ...
- ▶ Memory Instructions
 - ▶ `alloca`, `load`, `store`, ...

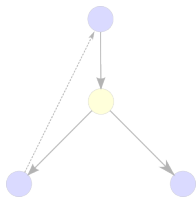
Complete Reference: <http://llvm.org/docs/LangRef.html>

LLVM to Symbolic Execution Graph

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

YES/NO/
MAYBE

?

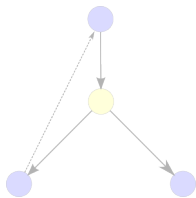


LLVM to Symbolic Execution Graph

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

YES/NO/
MAYBE

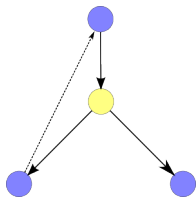


LLVM to Symbolic Execution Graph

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

YES/NO/
MAYBE

?



Problem Definition

Given: LLVM Program, Entry Point

Goal: Description of *at least all possible runs* from the entry point

Problem Definition

Given: LLVM Program, Entry Point

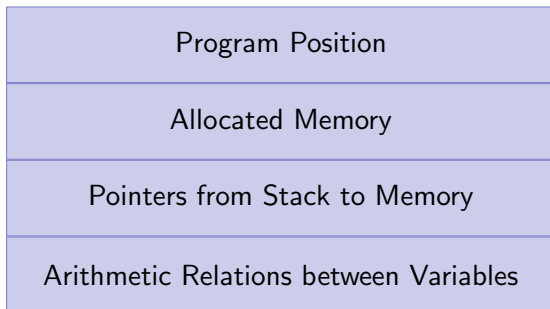
Goal: Description of *at least all possible runs* from the entry point

Idea: Abstract interpretation of program states

Abstract States

Program Position
Allocated Memory
Pointers from Stack to Memory
Arithmetic Relations between Variables

Abstract States



Entry Point: `define i32 strlen(i8* str) {`
 `c0 = load i8* str`
 `...`

Abstract States

Position:	(0, strlen)
Allocated Memory	
Pointers from Stack to Memory	
Arithmetic Relations between Variables	

Entry Point: `define i32 strlen(i8* str) {`
 `c0 = load i8* str`
 `...`

Abstract States

Position:	(0, strlen)
Alloc:	[alloc(str, v_end)]
Pointers from Stack to Memory	
Arithmetic Relations between Variables	

Entry Point: define i32 strlen(i8* str) {
 c0 = load i8* str
 ...

Abstract States

Position:	$(0, \text{strlen})$
Alloc:	$[\text{alloc}(\text{str}, v_{\text{end}})]$
Stack \rightarrow Memory:	$v_{\text{end}} \hookrightarrow 0$
Arithmetic Relations between Variables	

Entry Point: `define i32 strlen(i8* str) {`
 `c0 = load i8* str`
 `...`

Abstract States

Position:	$(0, \text{strlen})$
Alloc:	$[\text{alloc}(\text{str}, v_{\text{end}})]$
Stack \rightarrow Memory:	$v_{\text{end}} \hookrightarrow 0$
Relations:	\emptyset

Entry Point: `define i32 strlen(i8* str) {`
 `c0 = load i8* str`
 `...`

Evaluation of Abstract States

```
c0 = load i8* str
```

Position:
 $(0, \text{strlen})$

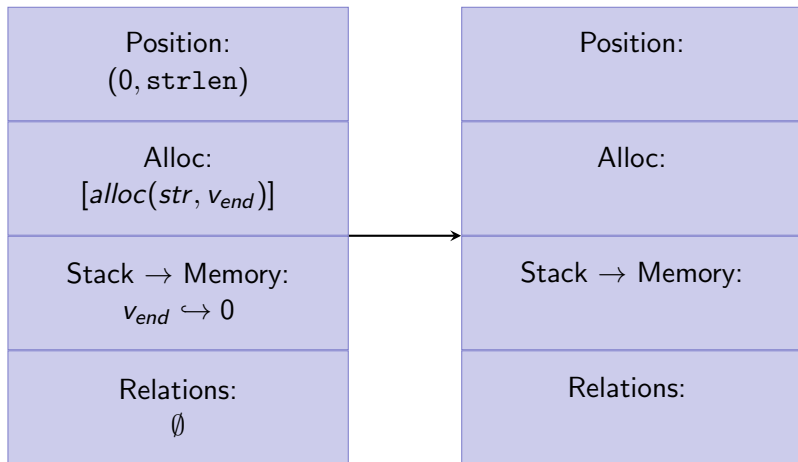
Alloc:
 $[\text{alloc}(\text{str}, v_{\text{end}})]$

Stack \rightarrow Memory:
 $v_{\text{end}} \hookrightarrow 0$

Relations:
 \emptyset

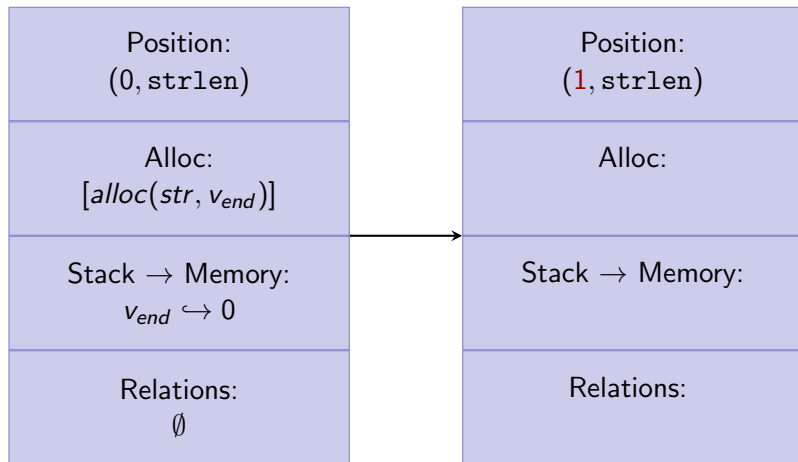
Evaluation of Abstract States

```
c0 = load i8* str
```



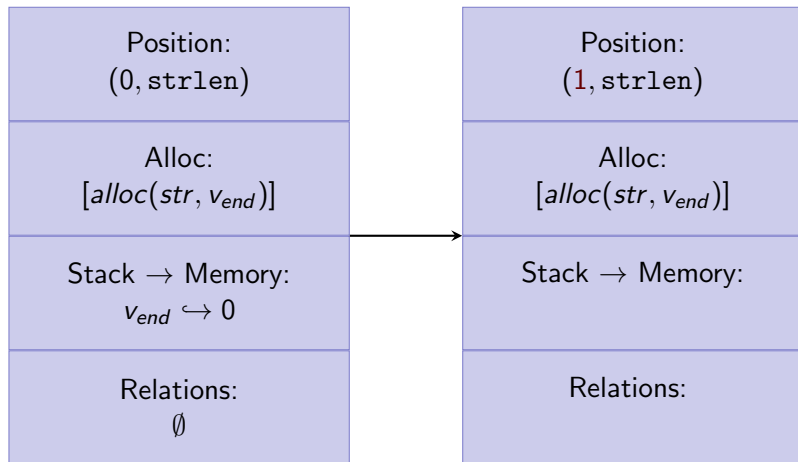
Evaluation of Abstract States

```
c0 = load i8* str
```



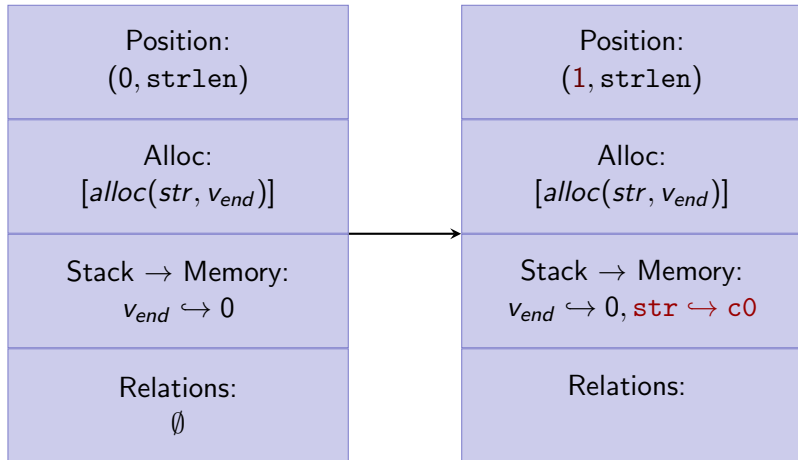
Evaluation of Abstract States

```
c0 = load i8* str
```



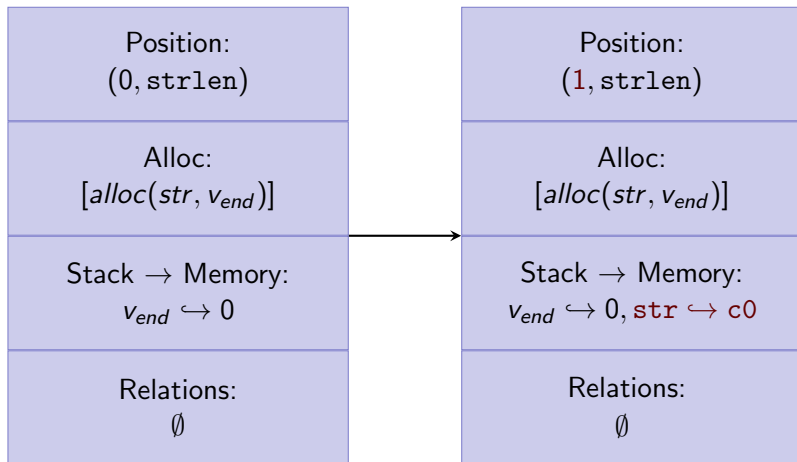
Evaluation of Abstract States

```
c0 = load i8* str
```



Evaluation of Abstract States

```
c0 = load i8* str
```



Refinement of Abstract States

```
c0zero = icmp eq i8 c0, 0
```

Position:
(1, strlen)

Alloc:
 $[alloc(str, v_{end})]$

Stack \rightarrow Memory:
 $v_{end} \hookrightarrow 0, str \hookrightarrow c0$

Relations:
 \emptyset

Refinement of Abstract States

`c0zero = $\underbrace{\text{icmp eq i8 c0, 0}}_{\text{c0==0}}$`

Position:
 $(1, \text{strlen})$

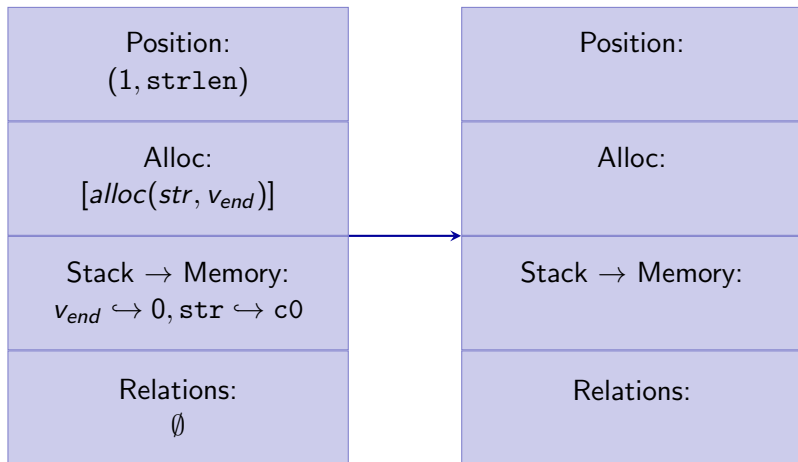
Alloc:
 $[\text{alloc}(\text{str}, v_{\text{end}})]$

Stack \rightarrow Memory:
 $v_{\text{end}} \hookrightarrow 0, \text{str} \hookrightarrow \text{c0}$

Relations:
 \emptyset

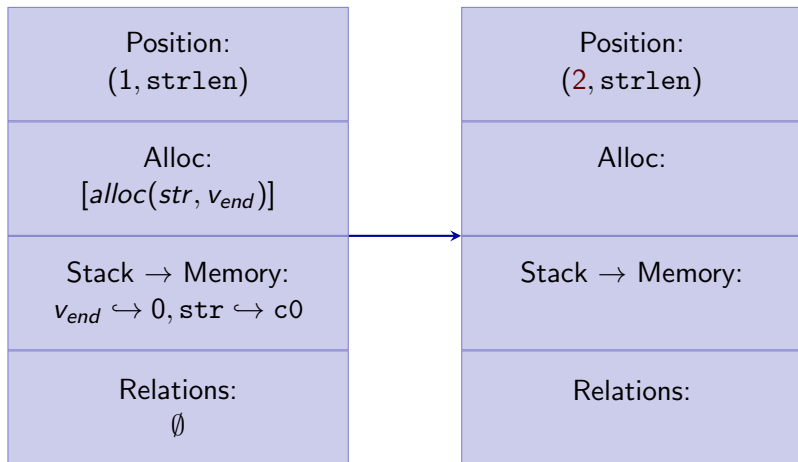
Refinement of Abstract States

`c0zero = $\underbrace{\text{icmp eq i8 c0, 0}}_{\text{c0==0}}$`



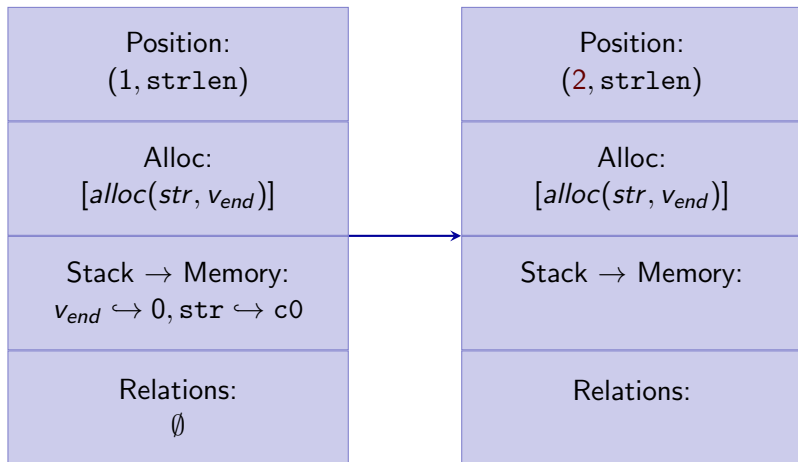
Refinement of Abstract States

`c0zero = $\underbrace{\text{icmp eq i8 c0, 0}}_{\text{c0==0}}$`

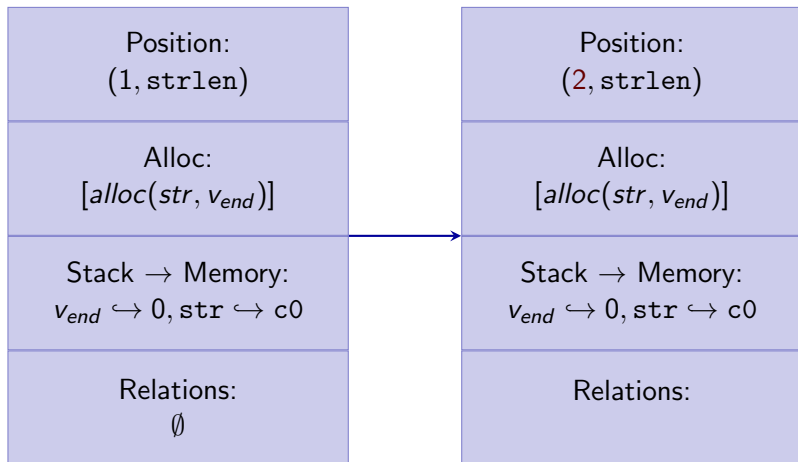


Refinement of Abstract States

`c0zero = $\underbrace{\text{icmp eq i8 c0, 0}}_{\text{c0==0}}$`

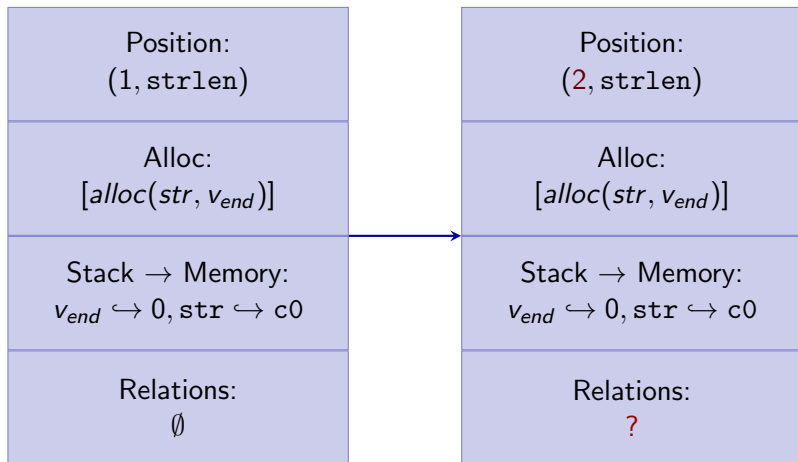


Refinement of Abstract States

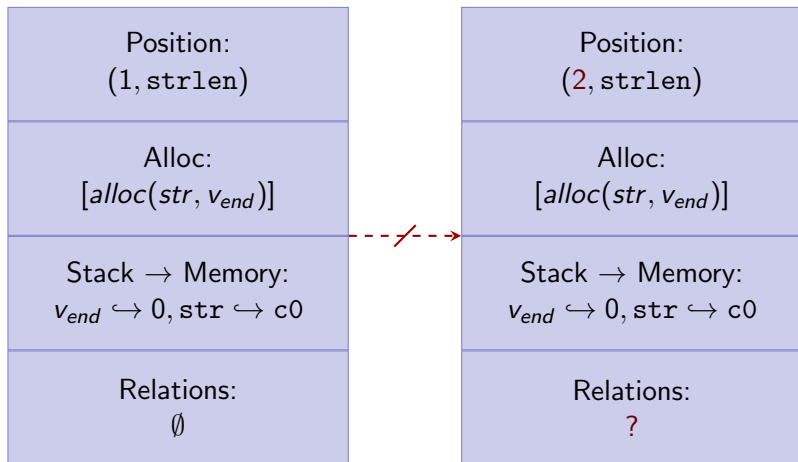
$$c0zero = \underbrace{\text{icmp eq i8 c0, 0}}_{c0==0}$$


Refinement of Abstract States

`c0zero = icmp eq i8 c0, 0`
 $c0==0$



Refinement of Abstract States

$$c0zero = \underbrace{icmp\ eq\ i8\ c0,\ 0}_{c0==0}$$


Refinement of Abstract States

```
c0zero = icmp eq i8 c0, 0
```

Position:
(1, strlen)

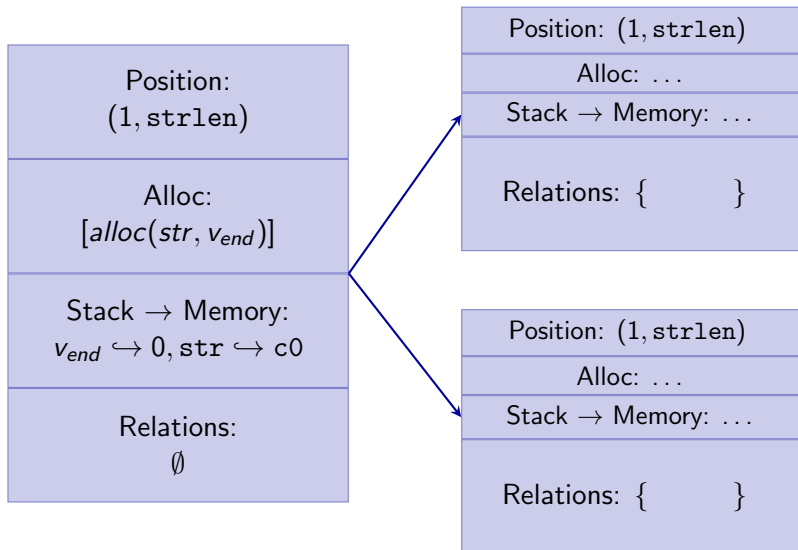
Alloc:
[*alloc*(*str*, *v_{end}*)]

Stack \rightarrow Memory:
 $v_{end} \hookrightarrow 0, str \hookrightarrow c0$

Relations:
 \emptyset

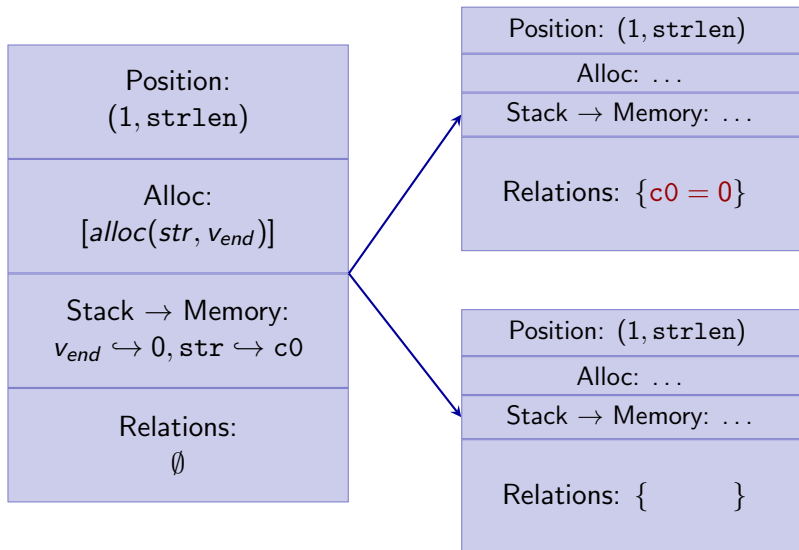
Refinement of Abstract States

```
c0zero = icmp eq i8 c0, 0
```



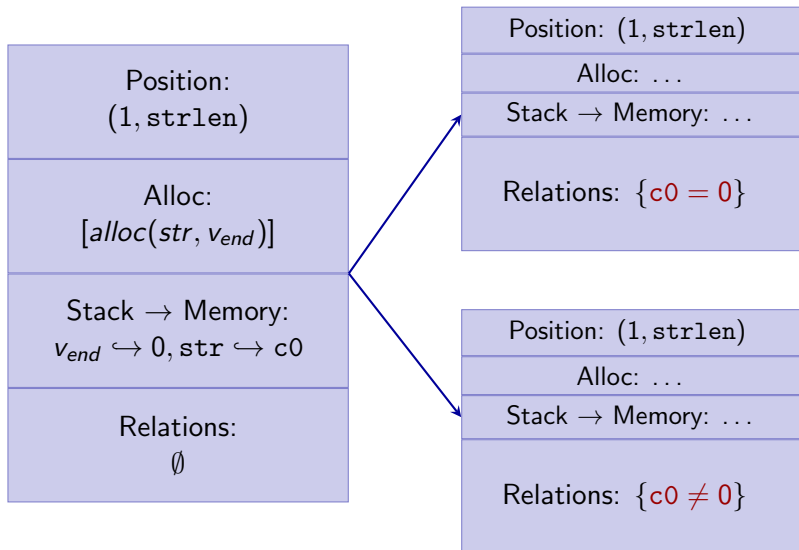
Refinement of Abstract States

```
c0zero = icmp eq i8 c0, 0
```



Refinement of Abstract States

```
c0zero = icmp eq i8 c0, 0
```



Merging of Abstract States

```
int strlen(char *str) {  
    char *s = str;  
    while(*s) s++;  
    return s-str;  
}
```

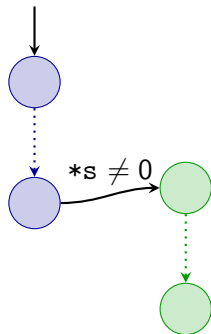
Merging of Abstract States

```
int strlen(char *str) {  
    char *s = str;  
    while(*s) s++;  
    return s-str;  
}
```



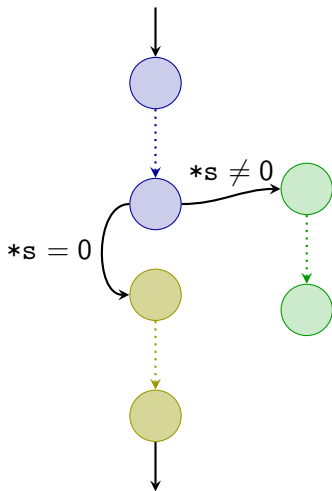
Merging of Abstract States

```
int strlen(char *str) {  
    char *s = str;  
    while(*s) s++;  
    return s-str;  
}
```



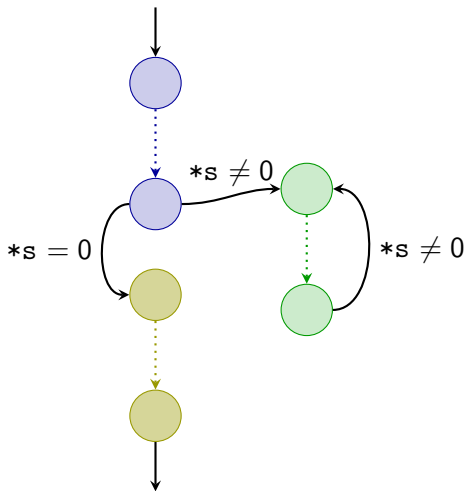
Merging of Abstract States

```
int strlen(char *str) {  
    char *s = str;  
    while(*s) s++;  
    return s-str;  
}
```



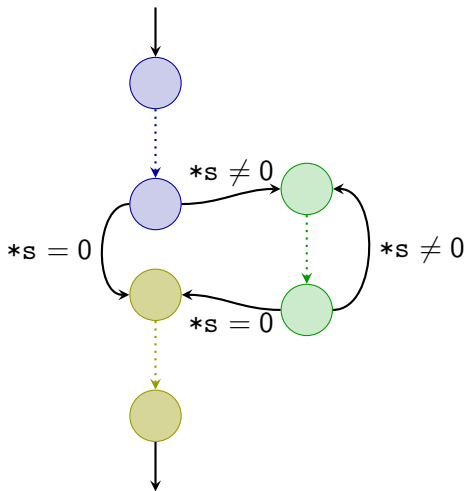
Merging of Abstract States

```
int strlen(char *str) {  
    char *s = str;  
    while(*s) s++;  
    return s-str;  
}
```

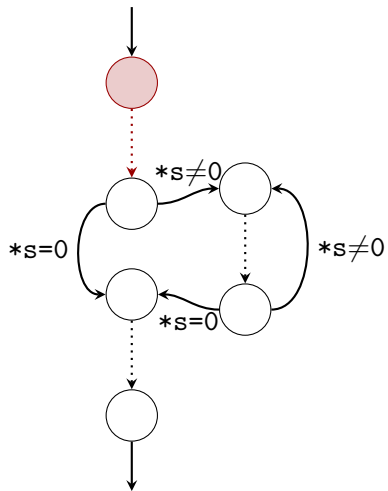


Merging of Abstract States

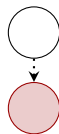
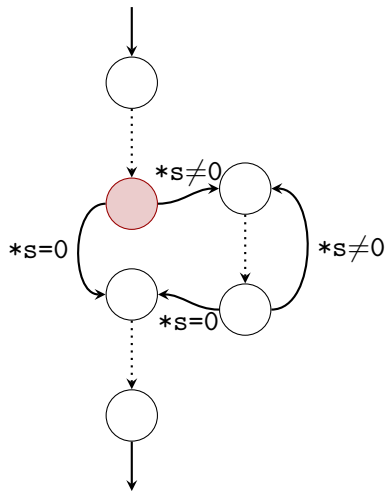
```
int strlen(char *str) {  
    char *s = str;  
    while(*s) s++;  
    return s-str;  
}
```



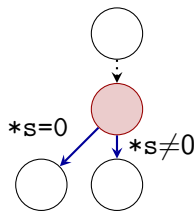
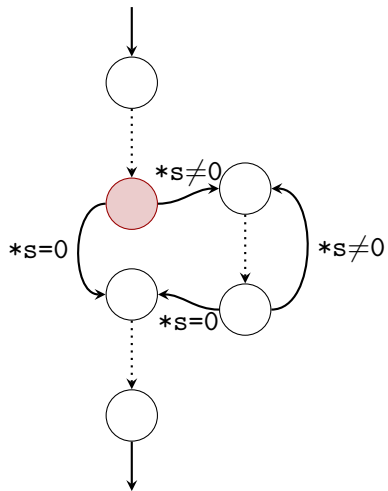
Merging of Abstract States



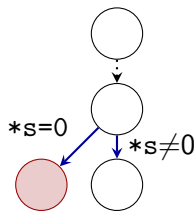
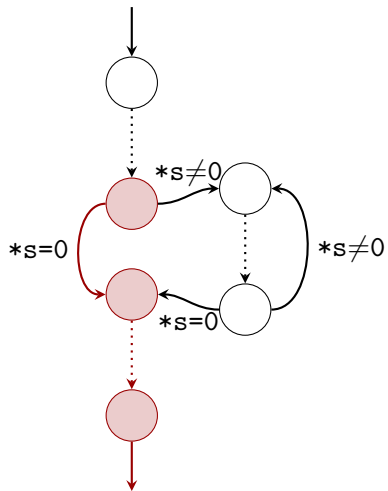
Merging of Abstract States



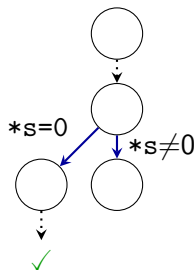
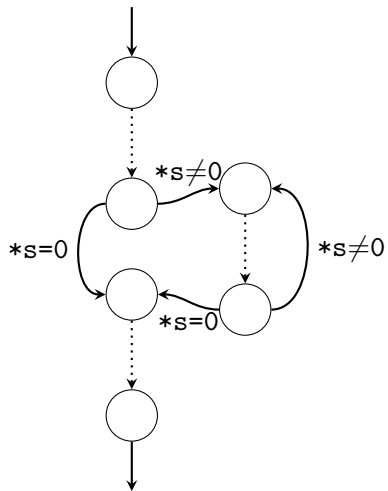
Merging of Abstract States



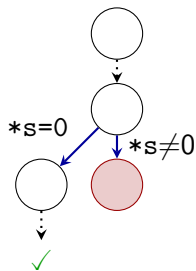
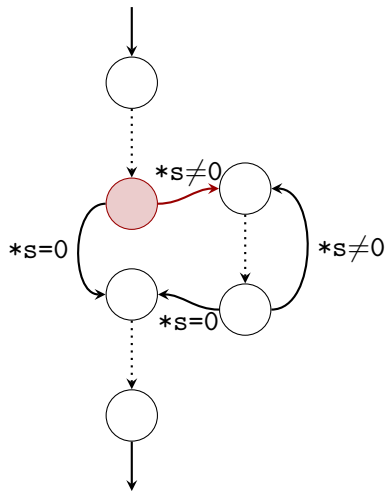
Merging of Abstract States



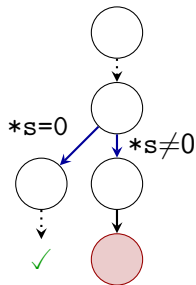
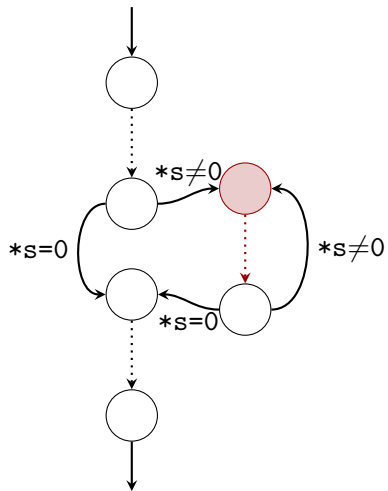
Merging of Abstract States



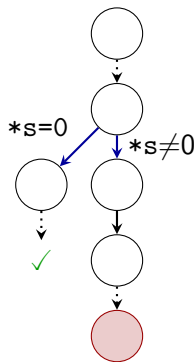
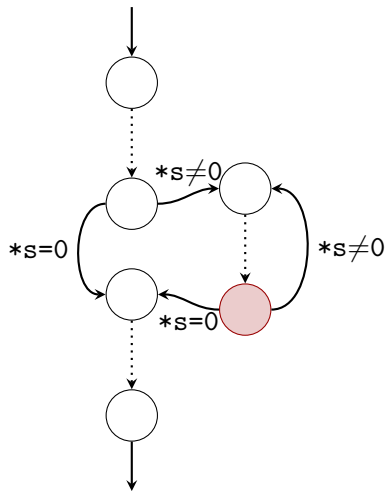
Merging of Abstract States



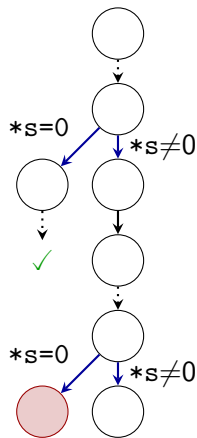
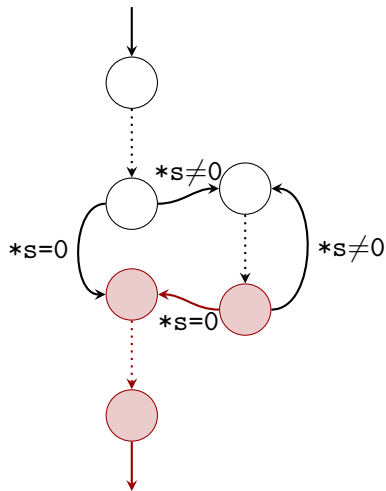
Merging of Abstract States



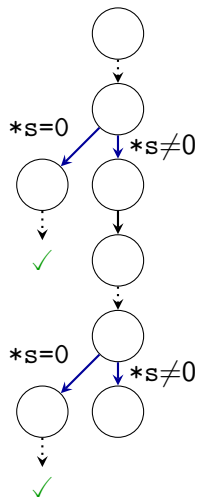
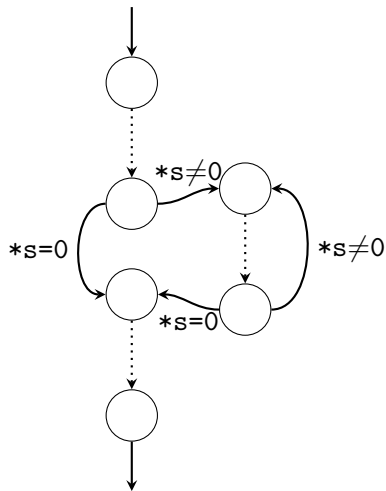
Merging of Abstract States



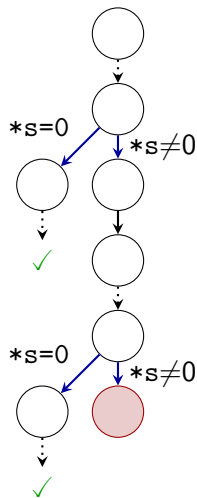
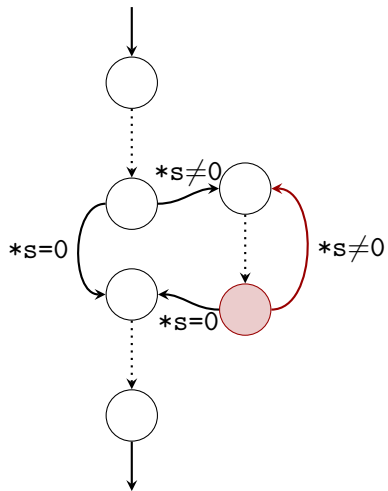
Merging of Abstract States



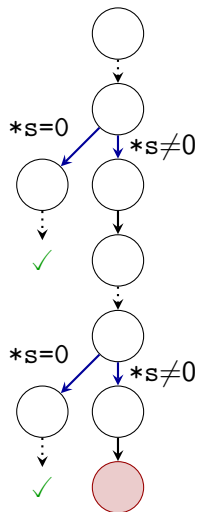
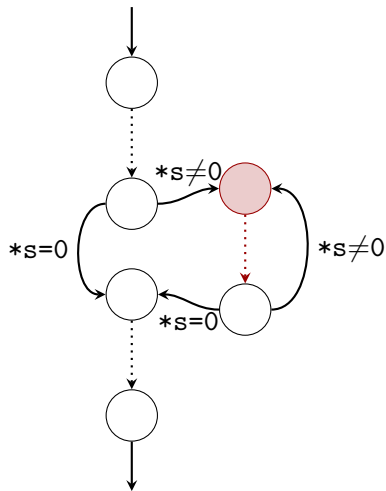
Merging of Abstract States



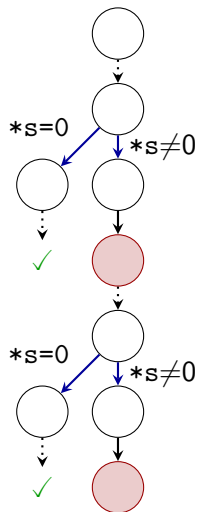
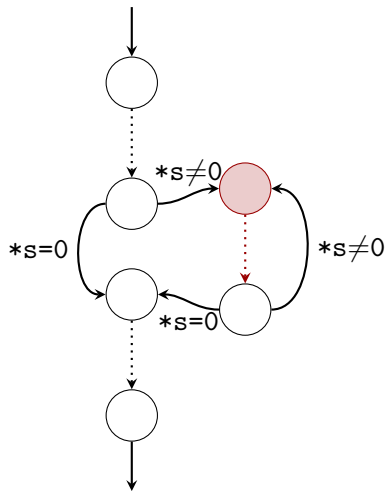
Merging of Abstract States



Merging of Abstract States



Merging of Abstract States

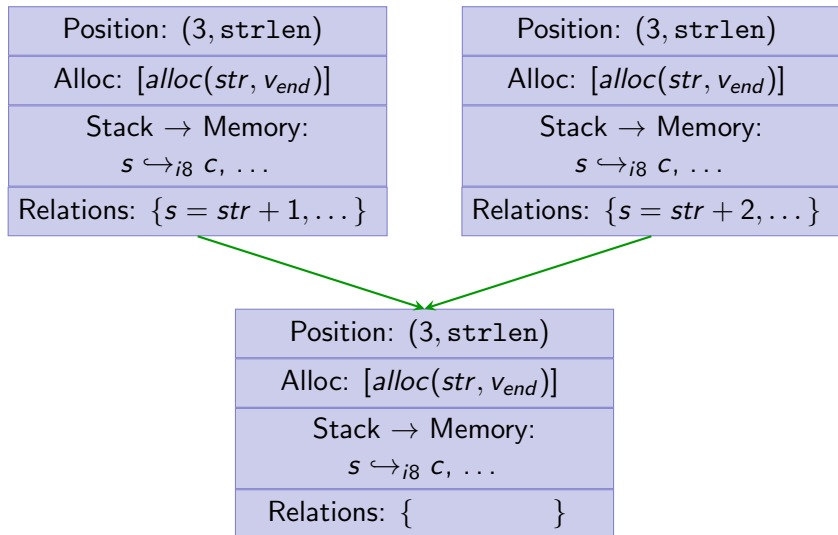


Merging of Abstract States

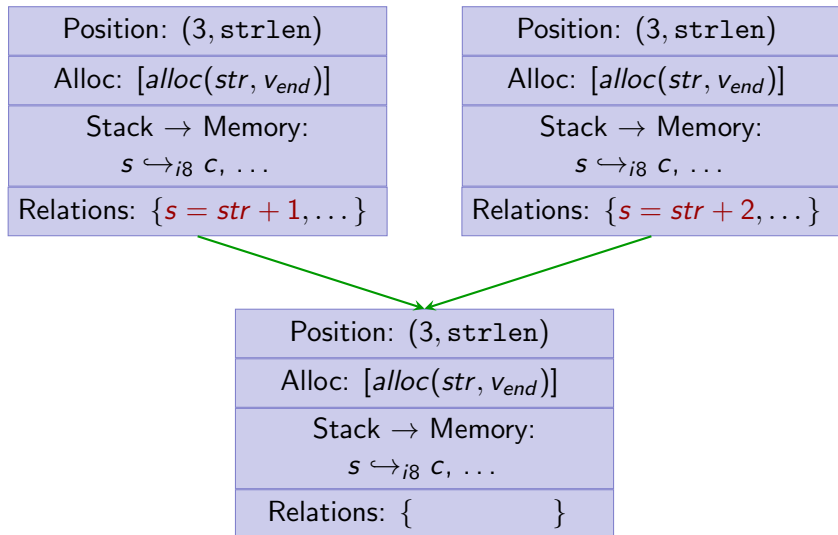
Position: $(3, \text{strlen})$
Alloc: $[\text{alloc}(\text{str}, v_{\text{end}})]$
Stack \rightarrow Memory: $s \hookrightarrow_{i8} c, \dots$
Relations: $\{s = \text{str} + 1, \dots\}$

Position: $(3, \text{strlen})$
Alloc: $[\text{alloc}(\text{str}, v_{\text{end}})]$
Stack \rightarrow Memory: $s \hookrightarrow_{i8} c, \dots$
Relations: $\{s = \text{str} + 2, \dots\}$

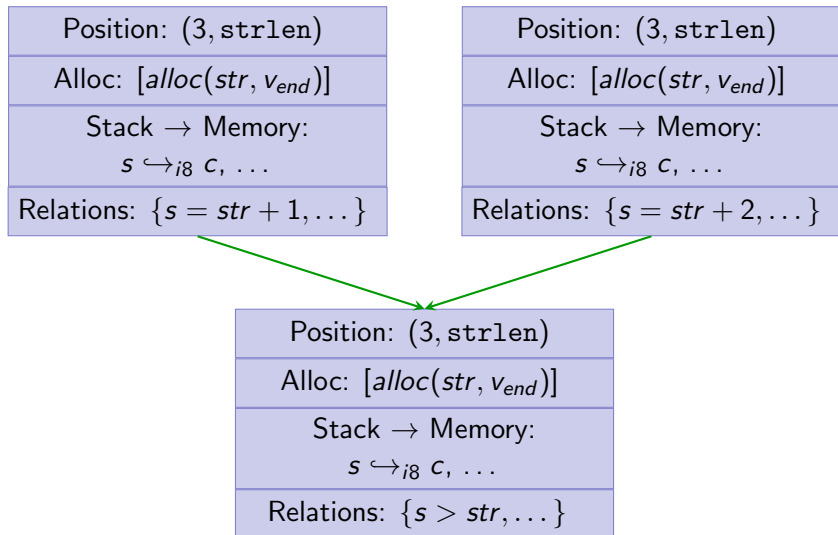
Merging of Abstract States



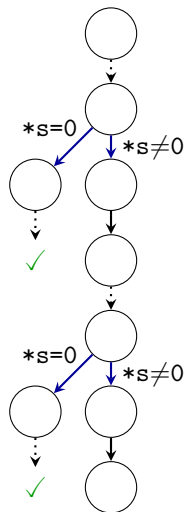
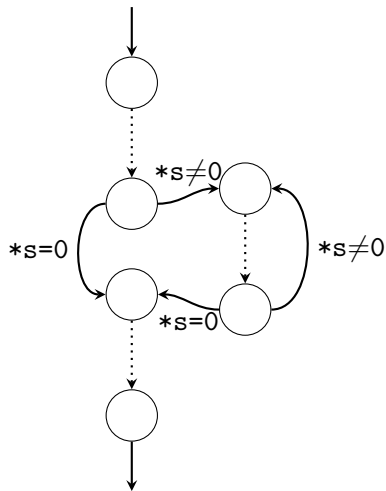
Merging of Abstract States



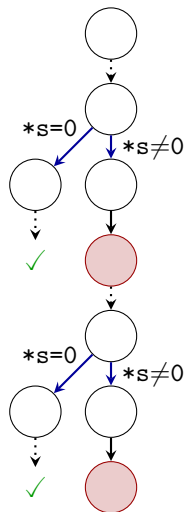
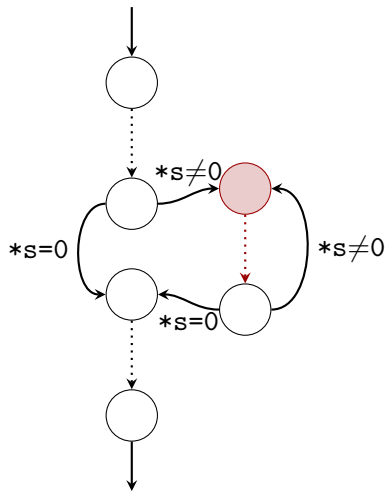
Merging of Abstract States



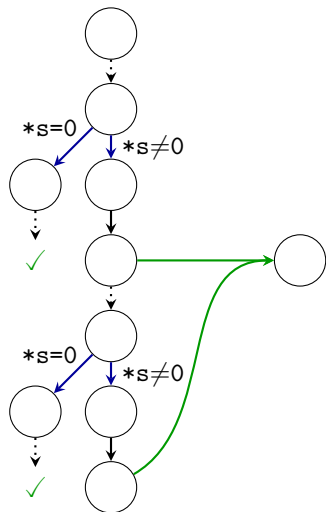
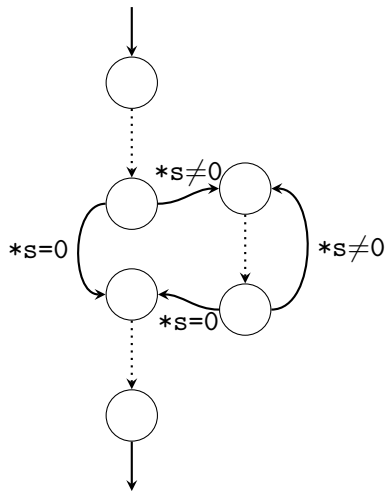
Merging of Abstract States



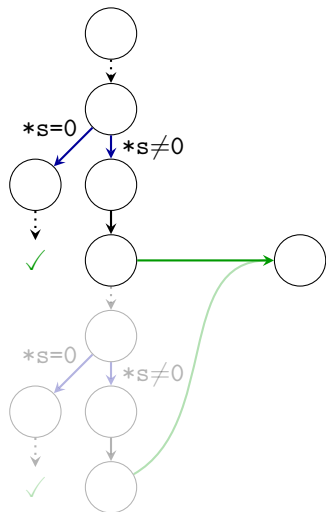
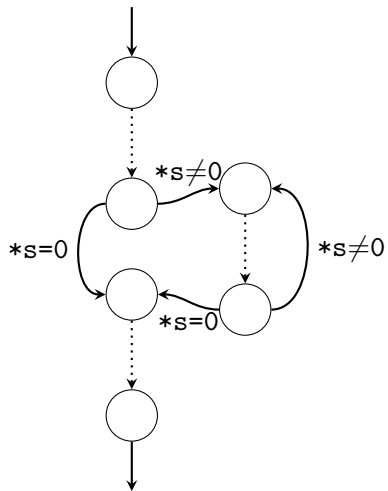
Merging of Abstract States



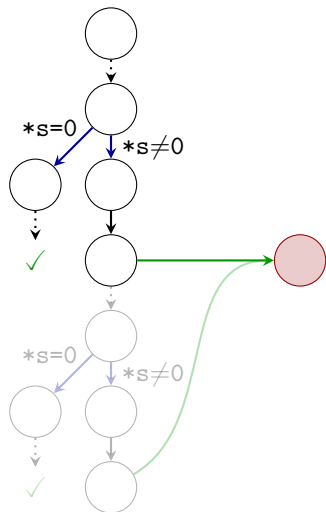
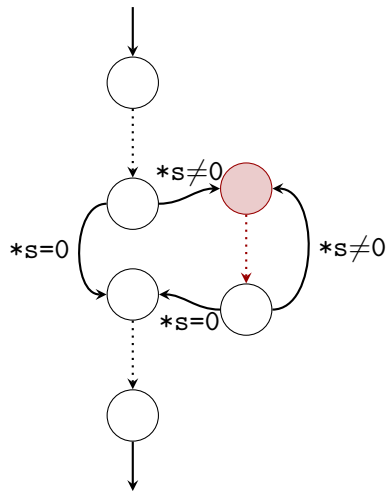
Merging of Abstract States



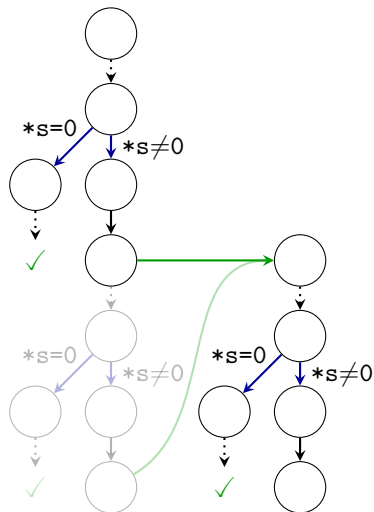
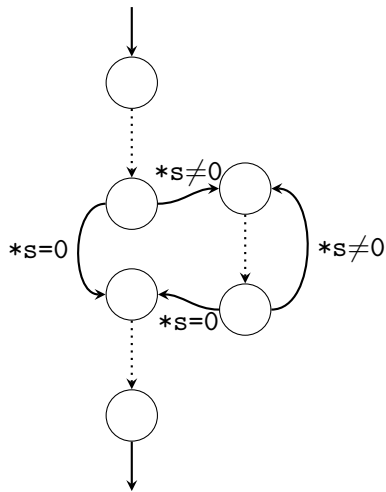
Merging of Abstract States



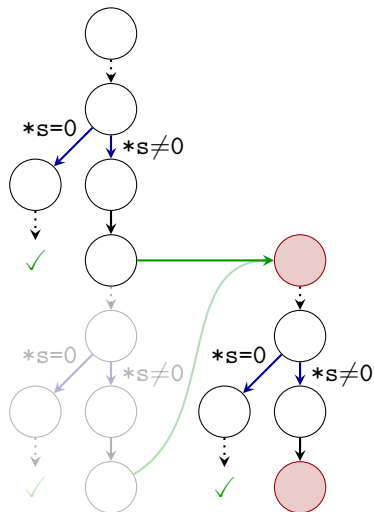
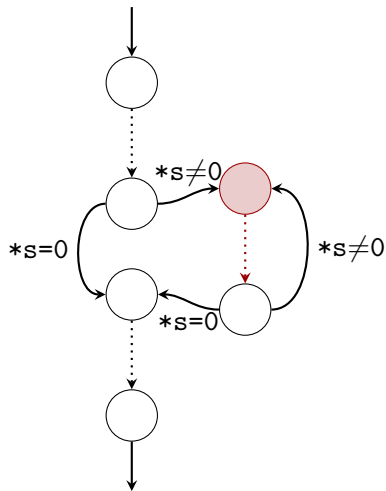
Merging of Abstract States



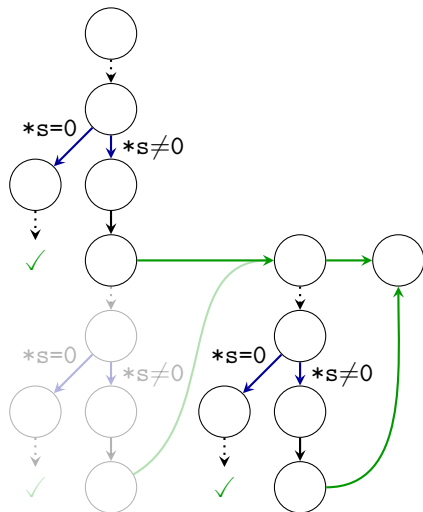
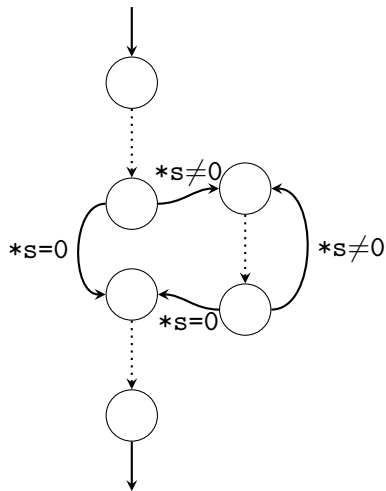
Merging of Abstract States



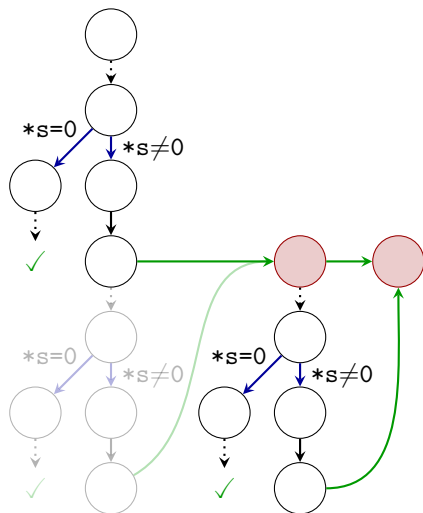
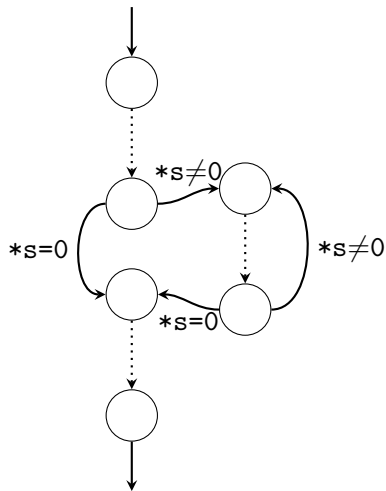
Merging of Abstract States



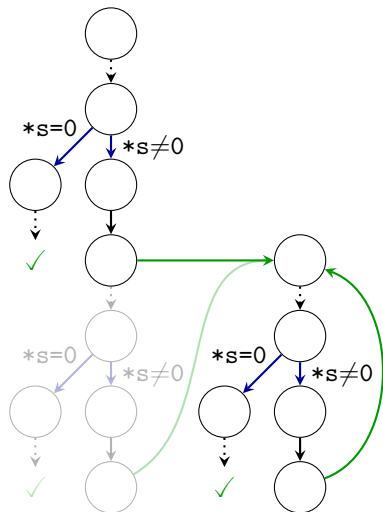
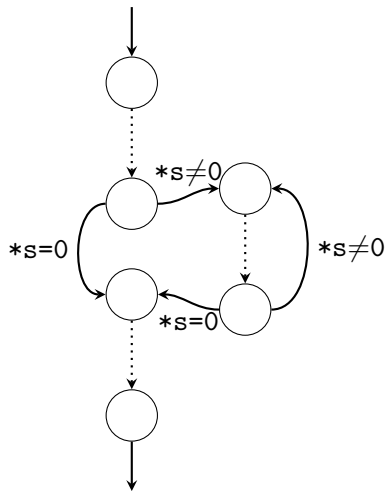
Merging of Abstract States



Merging of Abstract States



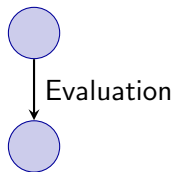
Merging of Abstract States



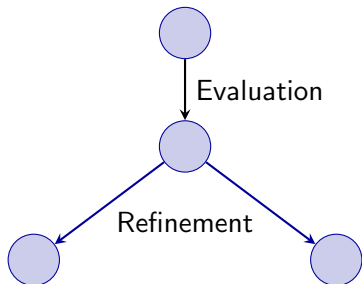
Summary: Symbolic Execution Graphs



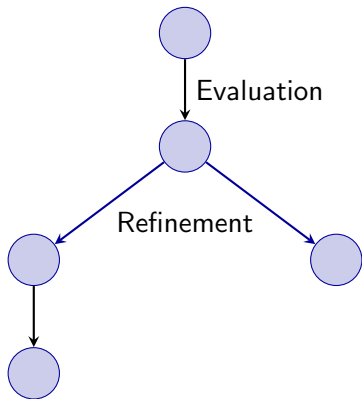
Summary: Symbolic Execution Graphs



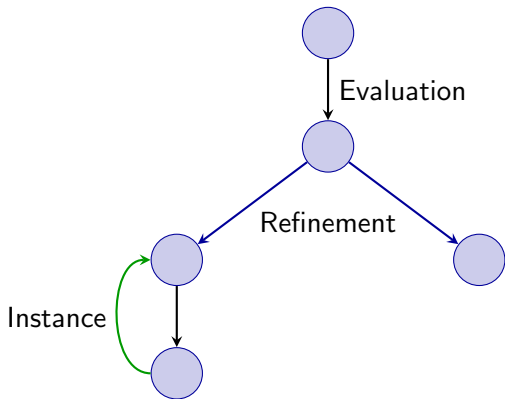
Summary: Symbolic Execution Graphs



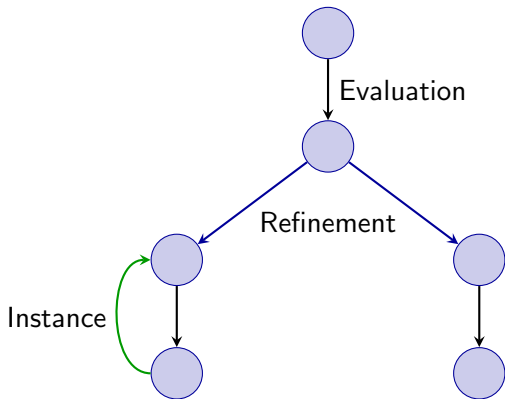
Summary: Symbolic Execution Graphs



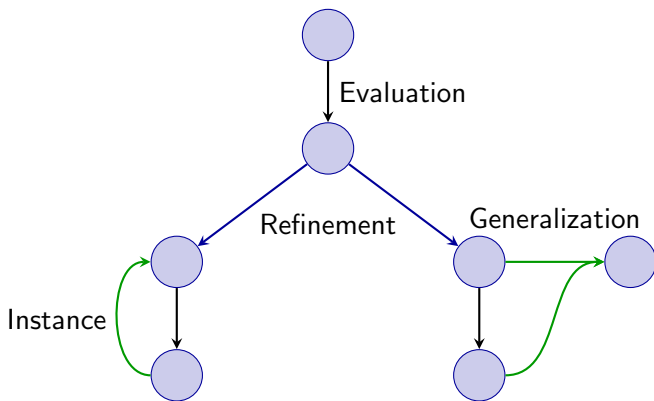
Summary: Symbolic Execution Graphs



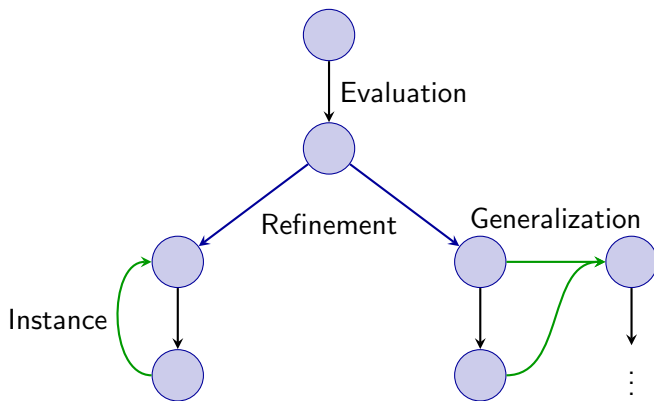
Summary: Symbolic Execution Graphs



Summary: Symbolic Execution Graphs



Summary: Symbolic Execution Graphs

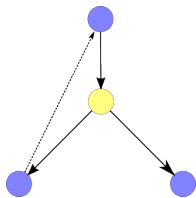


Symbolic Execution Graph to Integer Transition System

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



YES/NO/
MAYBE

?

$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

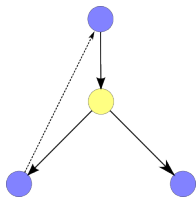
Integer Transition
System

Symbolic Execution Graph to Integer Transition System

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



YES/NO/
MAYBE

?

$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

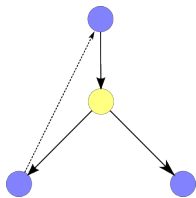
Integer Transition
System

Symbolic Execution Graph to Integer Transition System

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



YES/NO/
MAYBE

?

$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

Integer Transition Systems

Term Rewriting System:

$$f(x, y) \rightarrow g(y, x)$$

Integer Transition Systems

Term Rewriting System:

$$f(x, y) \rightarrow g(y, x)$$

Integer Transition System:

$$f(x, y) \rightarrow g(y + 1, x - 2)$$

Integer Transition Systems

Term Rewriting System:

$$f(x, y) \rightarrow g(y, x)$$

Integer Transition System:

$$f(x, y) \rightarrow g(y + 1, x - 2)$$

Integer Transition System:

$$f(x, y) \rightarrow g(y + 1, x - 2) \quad | \quad x > 0$$

Symbolic Execution Graph to Integer Transition System

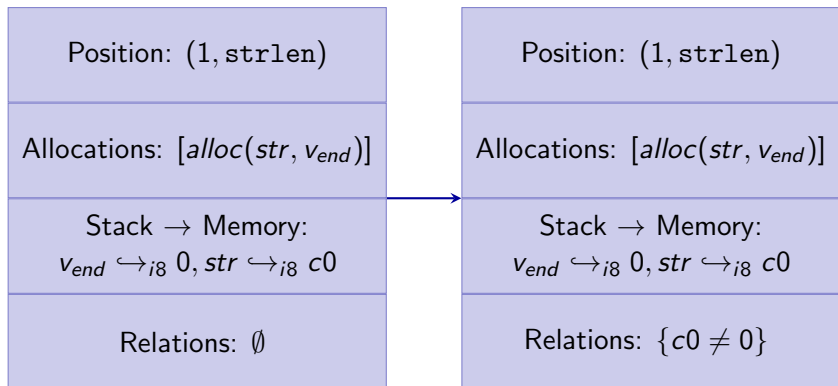
Position: $(1, \text{strlen})$

Allocations: $[\text{alloc}(\text{str}, v_{\text{end}})]$

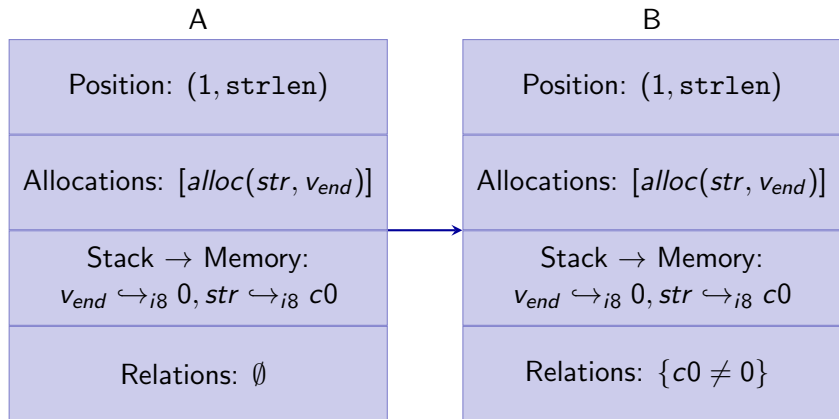
Stack \rightarrow Memory:
 $v_{\text{end}} \hookrightarrow_{i8} 0, \text{str} \hookrightarrow_{i8} c0$

Relations: \emptyset

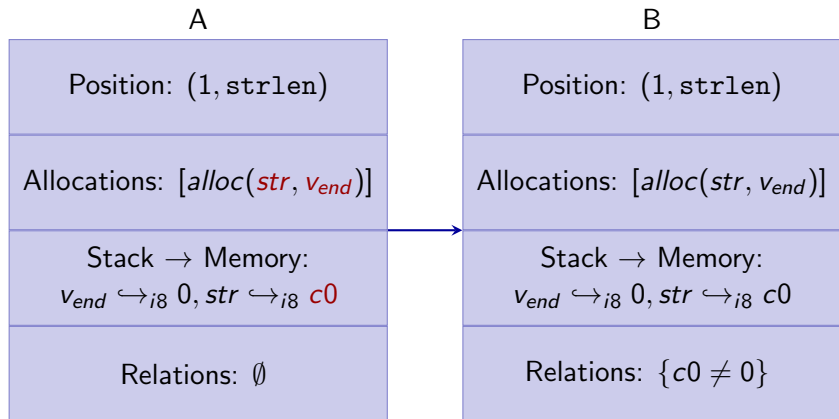
Symbolic Execution Graph to Integer Transition System



Symbolic Execution Graph to Integer Transition System

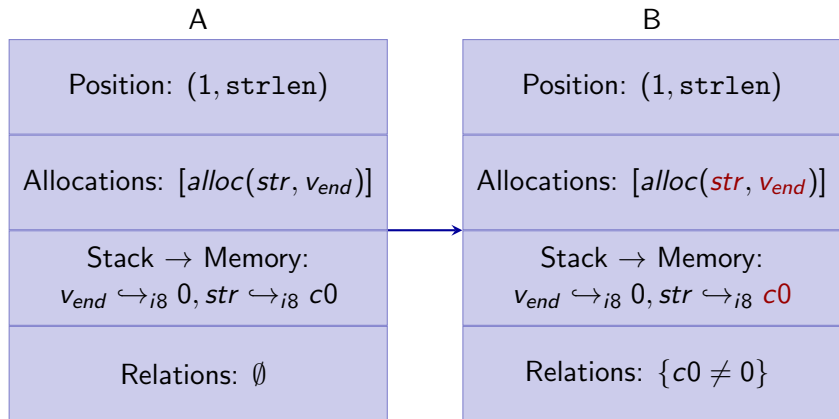


Symbolic Execution Graph to Integer Transition System



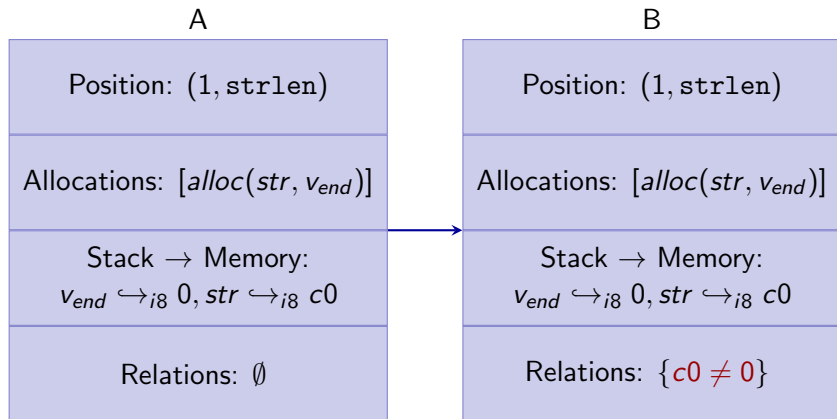
$$f_A(str, v_{end}, c0)$$

Symbolic Execution Graph to Integer Transition System



$$f_A(str, v_{end}, c0) \rightarrow f_B(str, v_{end}, c0)$$

Symbolic Execution Graph to Integer Transition System



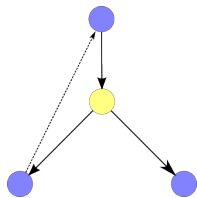
$$f_A(\text{str}, v_{\text{end}}, c0) \rightarrow f_B(\text{str}, v_{\text{end}}, c0) \mid c0 \neq 0$$

Termination of Integer Transition System

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

YES/NO/
MAYBE

?

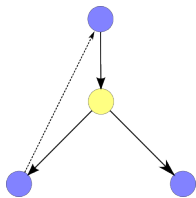


Termination of Integer Transition System

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



YES/NO/
MAYBE

?

$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

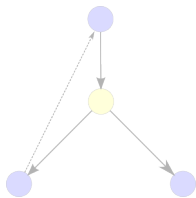
Integer Transition
System

Termination of Integer Transition System

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



YES/NO/
MAYBE



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

Termination of Integer Transition System

Well-studied problem

Termination of Integer Transition System

Well-studied problem

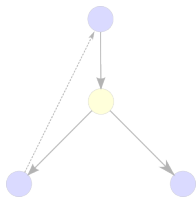
Use known techniques to show termination,
e.g. (Termination of Integer Term Rewriting, Fuhs et al., 2009)

Termination of Integer Transition System

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



YES/NO/
MAYBE



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

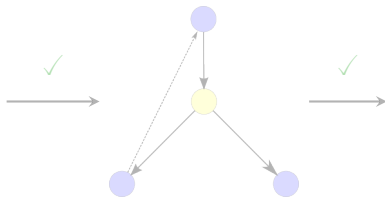
Integer Transition
System

Termination of Integer Transition System

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph

YES/NO/
MAYBE



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

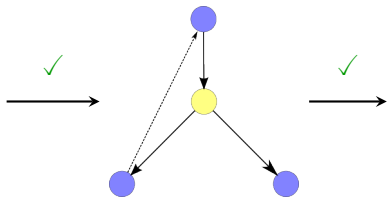
Integer Transition
System

Overview

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph

YES/NO/
MAYBE



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

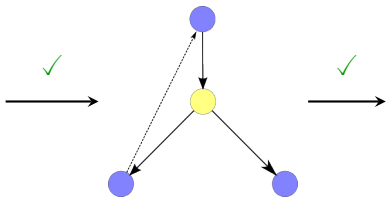
Integer Transition
System

My Contributions

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph

YES/NO/
MAYBE



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

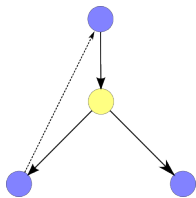
Integer Transition
System

My Contributions

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



YES/NO/
MAYBE



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

My Contributions

Given: Abstract State s , Integer Relation r

Question: Does $s \models r$?

My Contributions

- ▶ Reduction from state to set of arithmetic relations

My Contributions

- ▶ Reduction from state to set of arithmetic relations
- ▶ Inference of knowledge from states
 - ▶ Formulation of inference in terms of integer relations
 - ▶ New framework for inference
 - ▶ Parameterization of framework with abstract arithmetic domain
 - ▶ Formulation of existing inference in framework

My Contributions

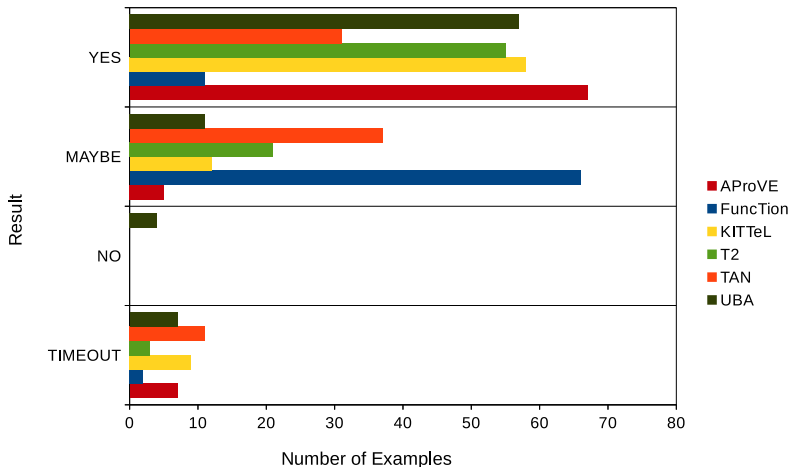
- ▶ Use of Octagon Domain for inference of relations
 - ▶ (The Octagon Abstract Domain, Miné, 2006)

My Contributions

- ▶ Use of Octagon Domain for inference of relations
 - ▶ (The Octagon Abstract Domain, Miné, 2006)
- ▶ Experimental comparison
 - ▶ “Traditional” inference
 - ▶ Inference in framework
 - ▶ New inference based on Octagons

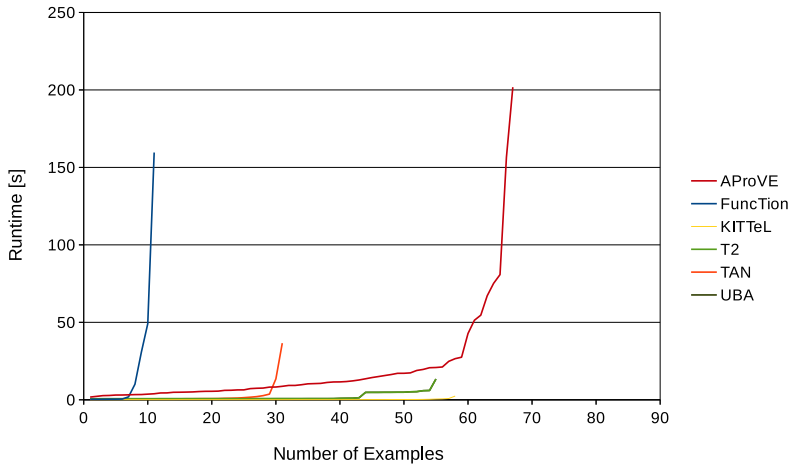
Evaluation: Empirical Evaluation

79 Integer Programs, Timeout: 300s



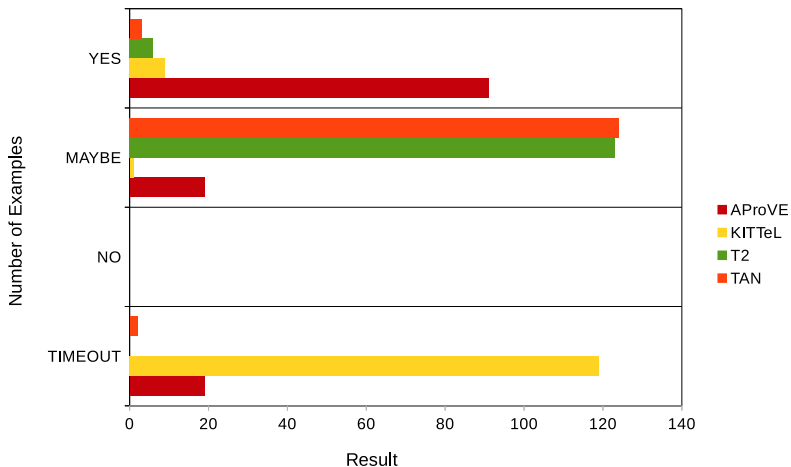
Evaluation: Empirical Evaluation

79 Integer Programs, Timeout: 300s



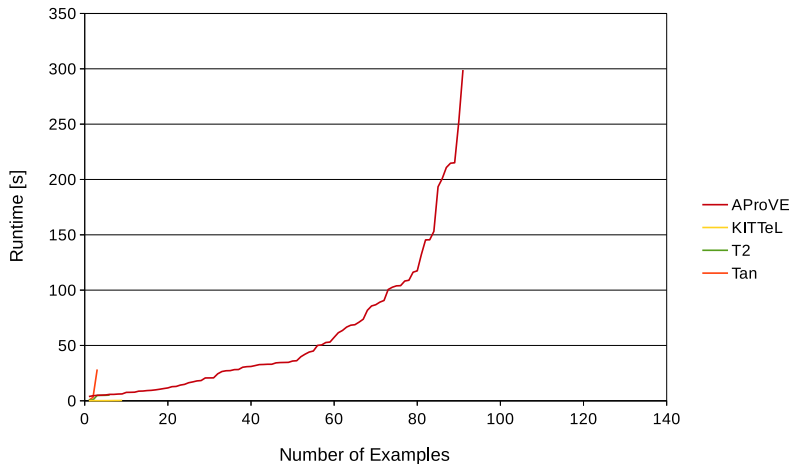
Evaluation: Empirical Evaluation

129 Pointer Programs, Timeout: 300s



Evaluation: Empirical Evaluation

129 Pointer Programs, Timeout: 300s



Complete evaluation at
<http://aprove.informatik.rwth-aachen.de/eval/Pointer/>

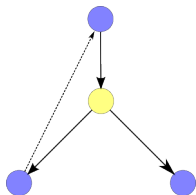
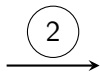
Thank you for your attention

`www.alexanderweinert.net`
`alexander.weinert@rwth-aachen.de`

THE
C
PROGRAMMING
LANGUAGE



LLVM



Symbolic Execution Graph



$f(x) \rightarrow g(x + 1)$
 $g(x) \rightarrow g(x - 1)$

Integer Transition
System

YES/NO/
MAYBE

