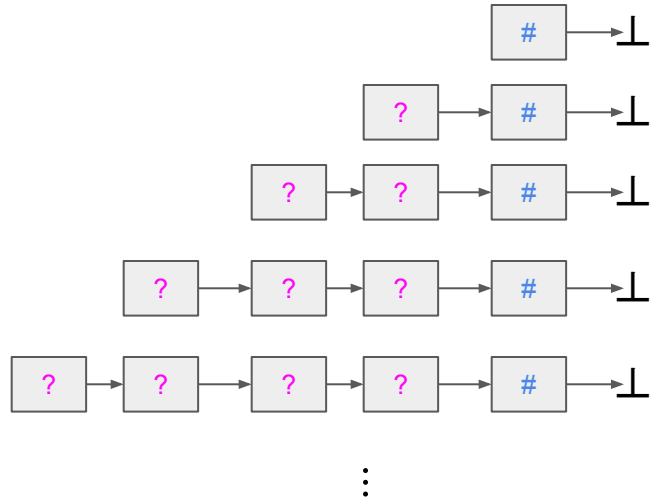
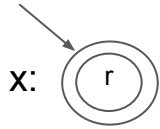


1. list* x = new { .next=NULL, .data=#};
2. while (nondet())
3. y = x; x = new { .next=y, .data=?};
4. while (x->data != #)
5. x = x->next; assert(x != NULL);

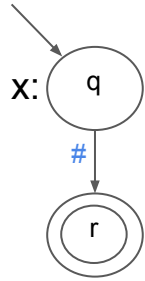




1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates

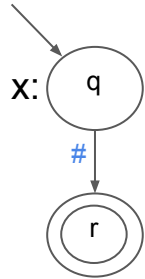
∅



1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates

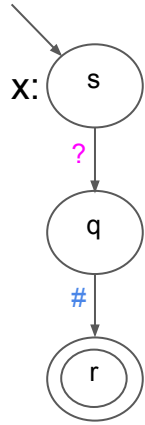
∅



1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates

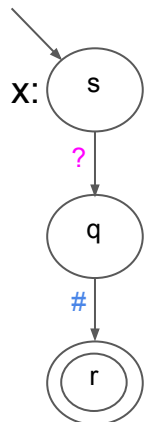
∅



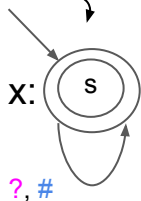
1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates

∅



Abstraction



1. list* x = new {.next=NULL, .data=#};
2. while (nondet())
3. y = x; x = new {.next=y, .data=?};
4. while (x->data != #)
5. x = x->next; assert(x != NULL);

Predicates

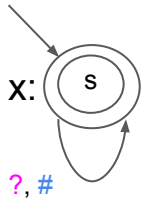
∅

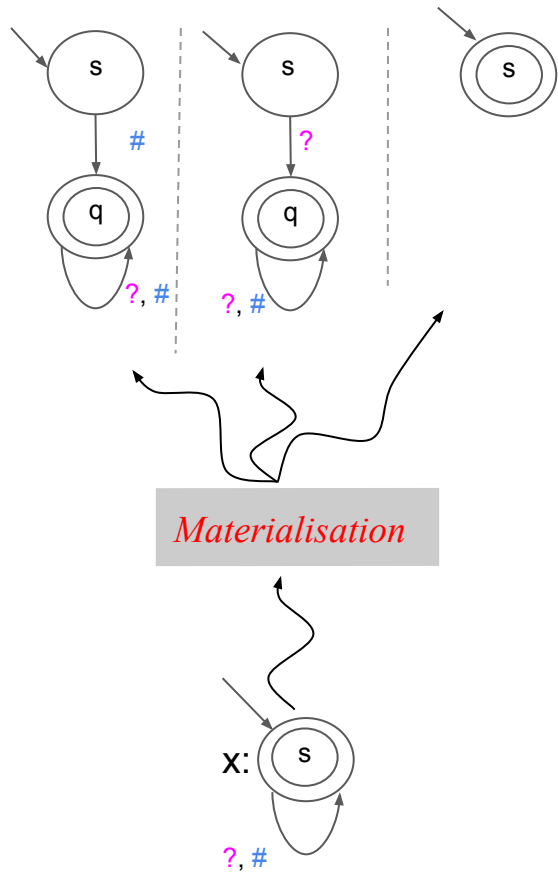
Predicate languages abstraction collapses states having languages with non-empty intersection with the same set of predicate languages (also represented by automata).

Predicates

∅

1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

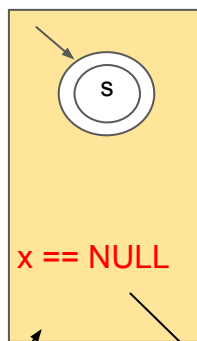
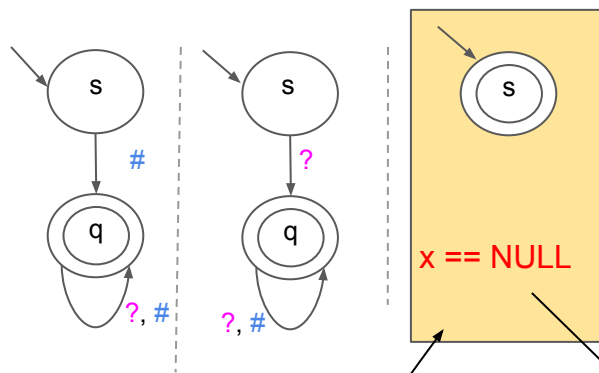




1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates

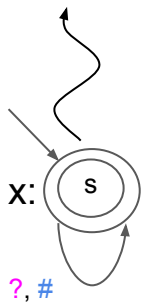
\emptyset



1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

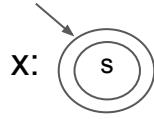
ERROR: Invalid dereference

Materialisation



Predicates

\emptyset



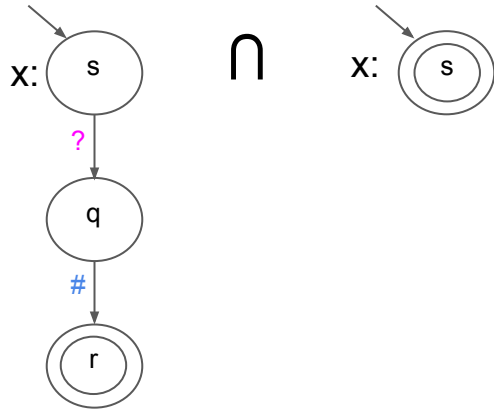
Validate Counterexample

1. list* x = new {.next=NULL, .data=#};
2. while (nondet())
3. y = x; x = new {.next=y, .data=?};
4. while (x->data != #)
5. x = x->next; assert(x != NULL);

Predicates

∅

Forward automaton Backward automaton

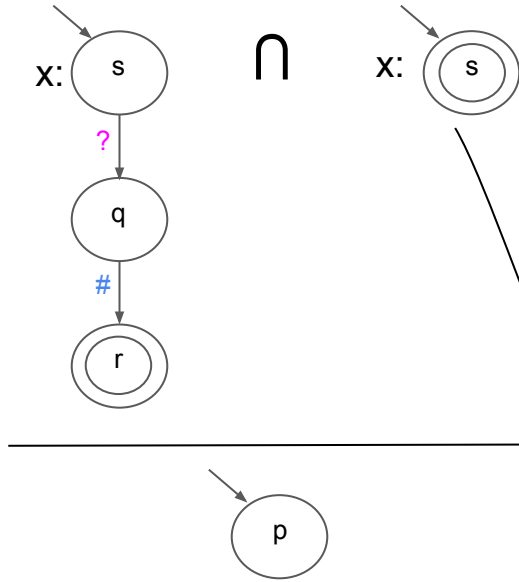


1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates

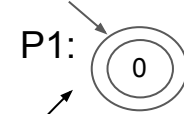
\emptyset

Forward automaton Backward automaton



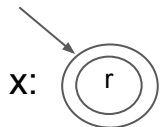
```
1. list* x = new {.next=NULL, .data=#};  
2. while (nondet())  
3.   y = x; x = new {.next=y, .data=?};  
  
4. while (x->data != #)  
5.   x = x->next; assert(x != NULL);
```

Predicates



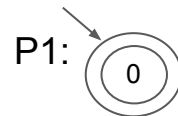
Interpolating path to CE and
obtaining a new predicate

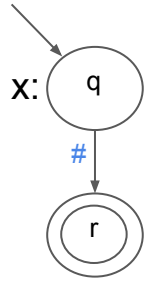
Restarting the analysis



1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

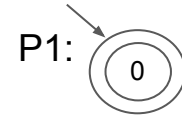
Predicates

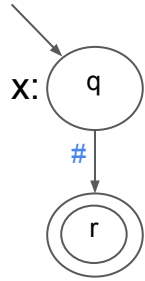




1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

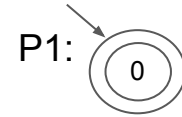
Predicates

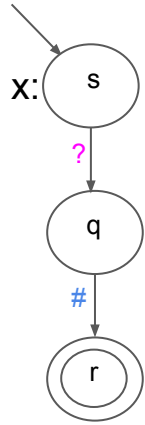




1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

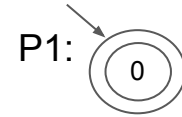
Predicates

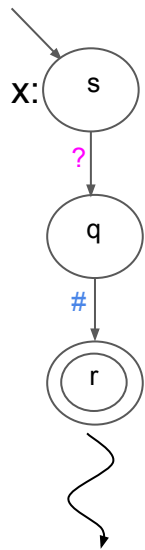




1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates

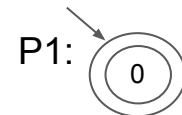




Abstraction

1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates

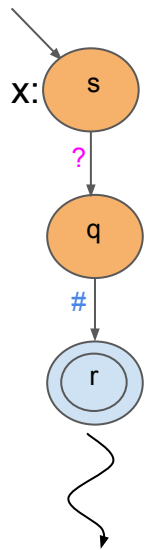


Intersection

$$L(P1) \cap L(s) = \emptyset$$

$$L(P1) \cap L(q) = \emptyset$$

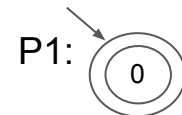
$$L(P1) \cap L(r) = \{\epsilon\}$$



Abstraction

1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates

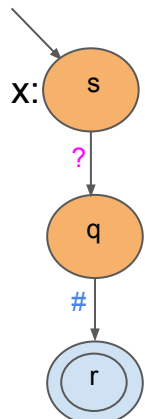


Intersection

$$L(P1) \cap L(s) = \emptyset$$

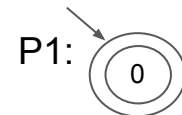
$$L(P1) \cap L(q) = \emptyset$$

$$L(P1) \cap L(r) = \{\epsilon\}$$

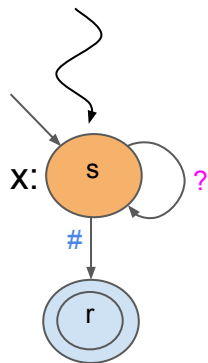


1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates



Abstraction



Intersection

$$L(P1) \cap L(s) = \emptyset$$

$$L(P1) \cap L(q) = \emptyset$$

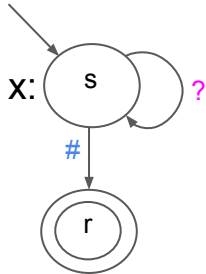
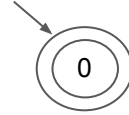
$$L(P1) \cap L(r) = \{\epsilon\}$$

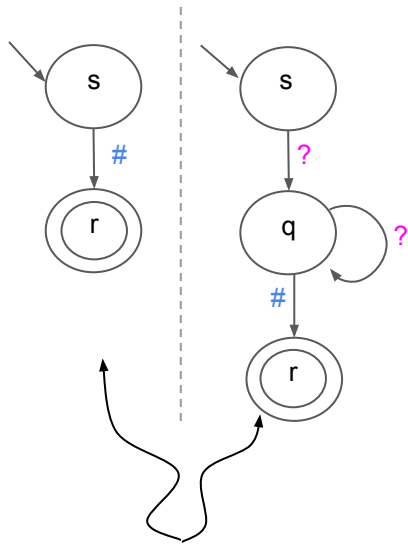
```

1. list* x = new {.next=NULL, .data=#};
2. while (nondet())
3.     y = x; x = new {.next=y, .data=?};
4. while (x->data != #)
5.     x = x->next; assert(x != NULL);

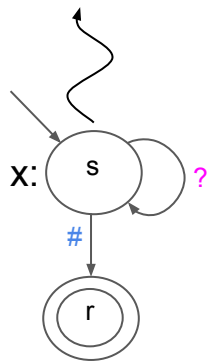
```

Predicates





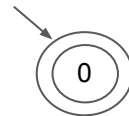
Materialisation

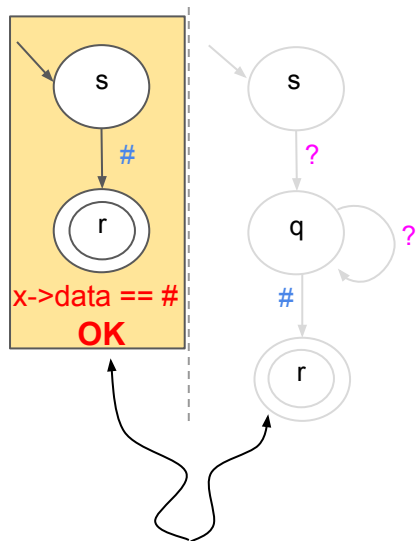


1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

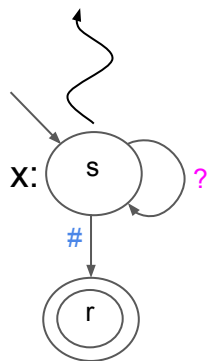
Valid dereference - exluded CE

Predicates



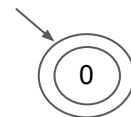


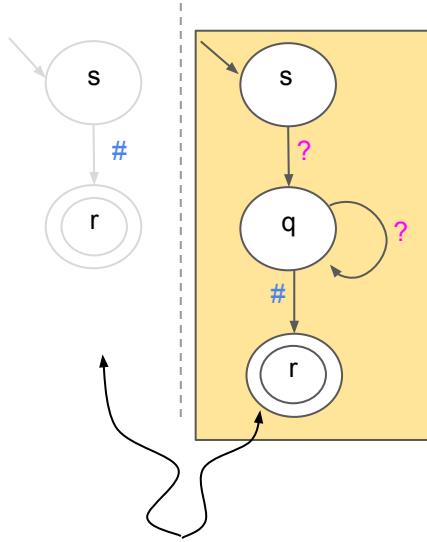
Materialisation



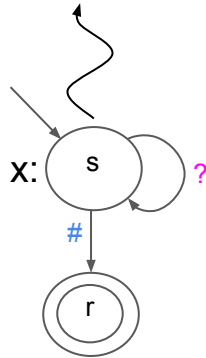
1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates



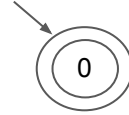


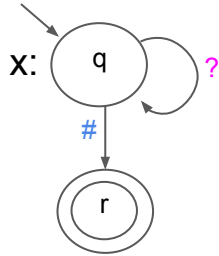
Materialisation



1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

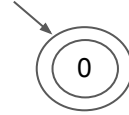
Predicates

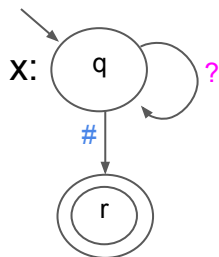




1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

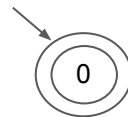
Predicates





1. `list* x = new {.next=NULL, .data=#};`
2. `while (nondet())`
3. `y = x; x = new {.next=y, .data=?};`
4. `while (x->data != #)`
5. `x = x->next; assert(x != NULL);`

Predicates



Program is safe

END