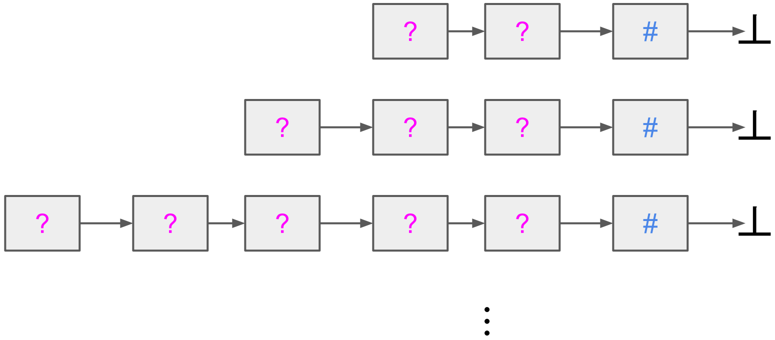
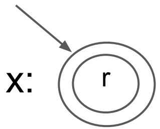
```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
   x = x->next; assert(x != NULL);
```

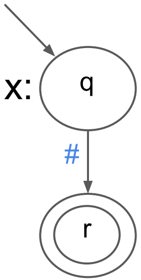


```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

∅

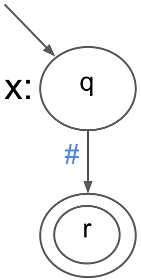




```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

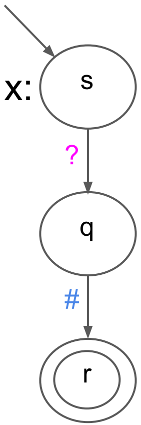
∅



```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

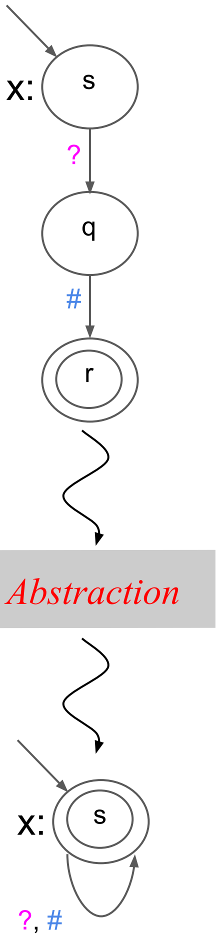
∅



```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

∅



```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

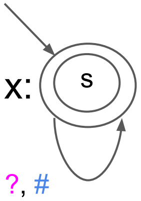
∅

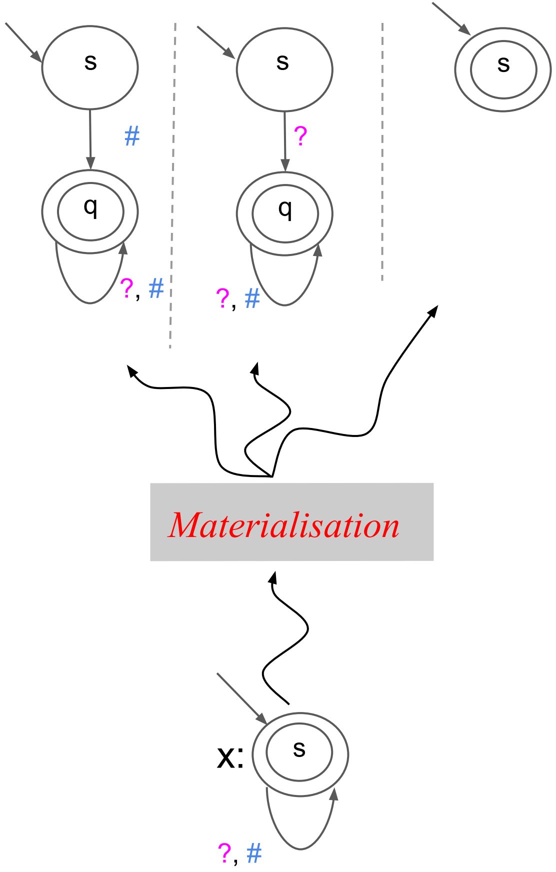
Predicate abstraction collapses states with non-empty intersection with the same set of predicates

```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

∅



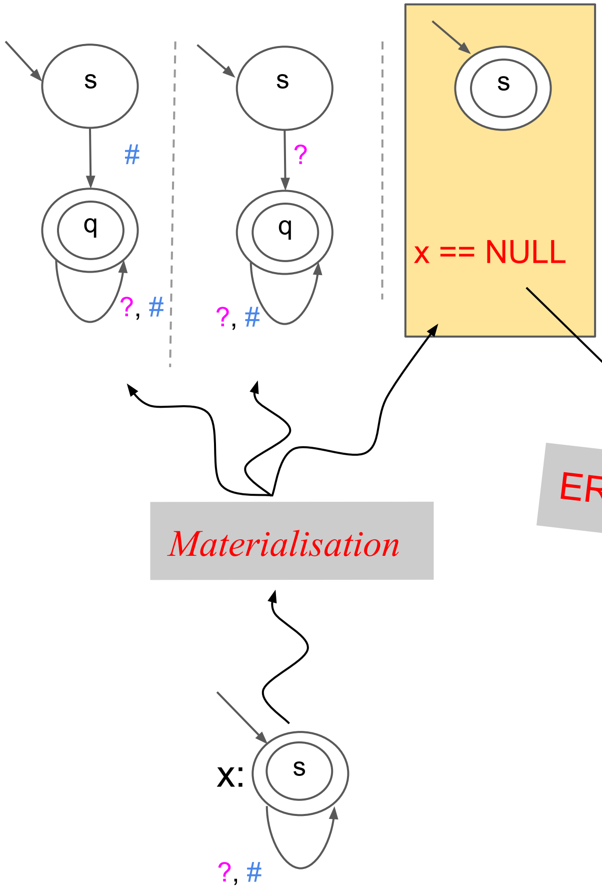


Materialisation

```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

∅



x == NULL

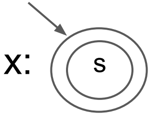
Materialisation

```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

∅

ERROR: Invalid dereference



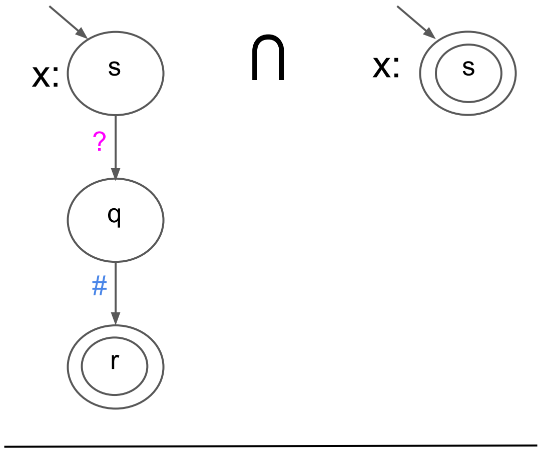
Validate Counterexample

```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

∅

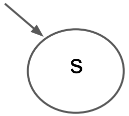
Forward automaton Backward automaton



```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
   x = x->next; assert(x != NULL);
```

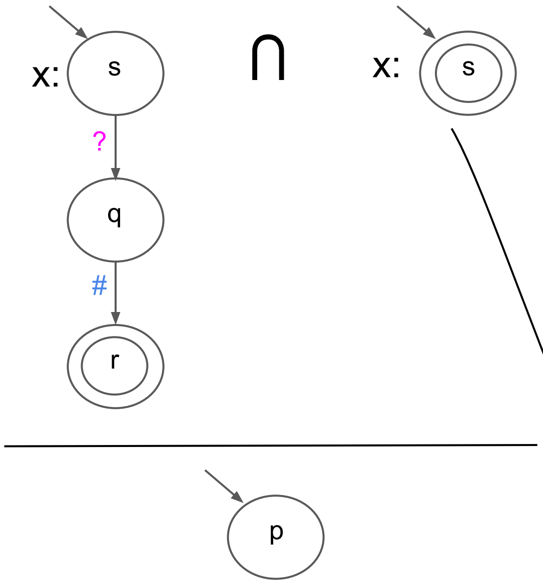
Predicates

∅



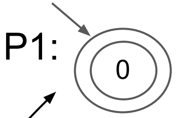
Empty Language - Spurious CE

Forward automaton Backward automaton



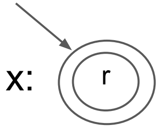
```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates



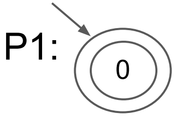
Interpolating new predicate
(and renaming states)

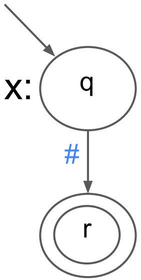
Restarting the analysis



```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

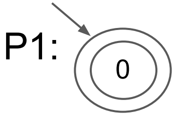
Predicates

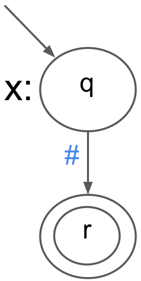




```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

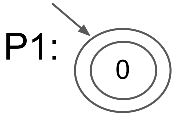
Predicates

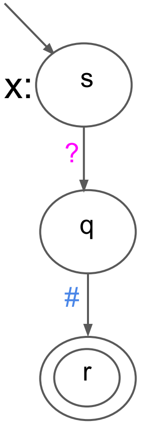




```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

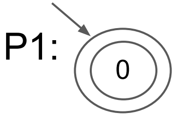
Predicates

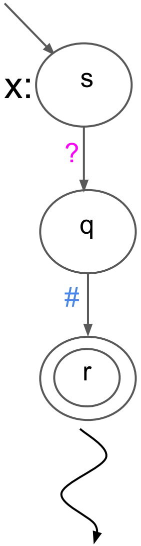




```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

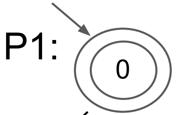




```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Abstraction

Predicates

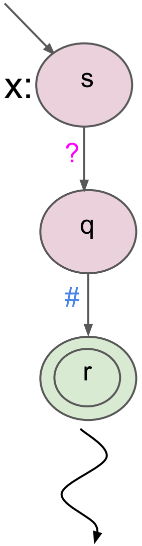


Intersection

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

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

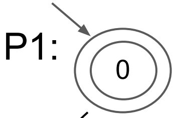
$L(P1) \cap L(r) = \{\varepsilon\}$



Abstraction

```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

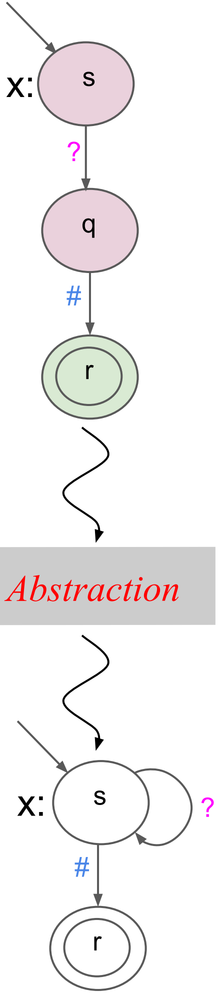


Intersection

L(P1) ∩ L(s) = ∅

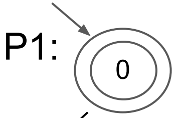
L(P1) ∩ L(q) = ∅

L(P1) ∩ L(r) = {ε}



```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates



Intersection

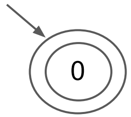
L(P1) ∩ L(s) = ∅

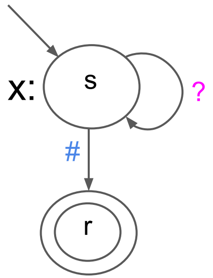
L(P1) ∩ L(q) = ∅

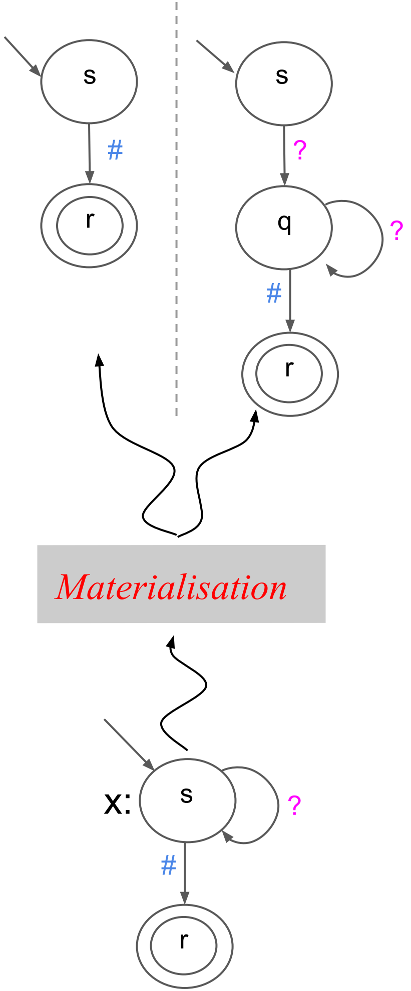
L(P1) ∩ L(r) = {ε}

```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
   x = x->next; assert(x != NULL);
```

Predicates





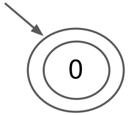


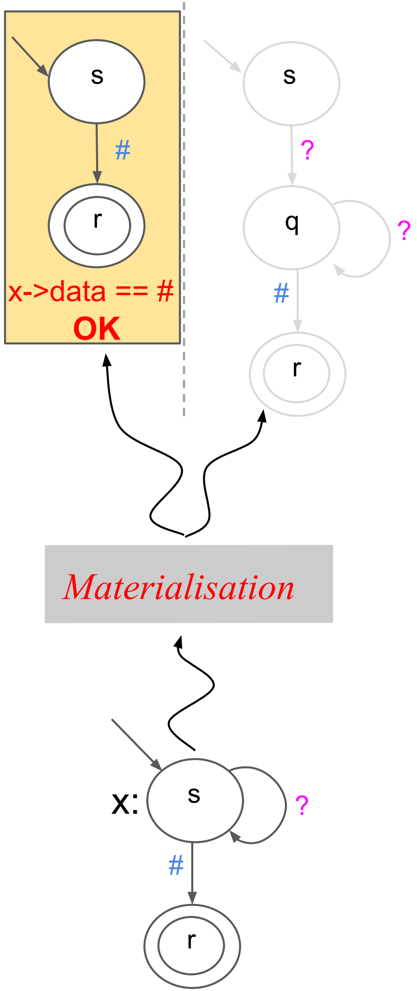
Materialisation

```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Valid dereference - exluded CE

Predicates





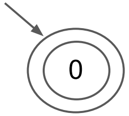
x->data == #

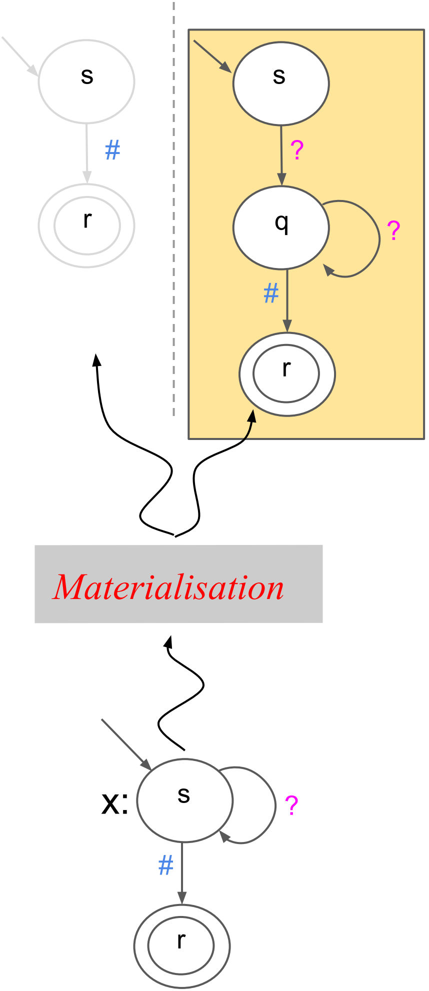
OK

Materialisation

```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates

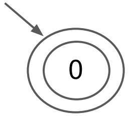


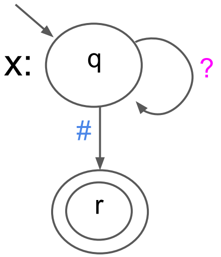


Materialisation

```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
   x = x->next; assert(x != NULL);
```

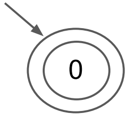
Predicates

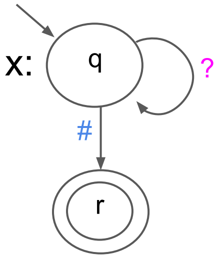




```
list* x = new {.next=NULL, .data=#};
while (nondet())
    y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

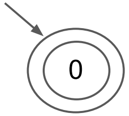
Predicates





```
list* x = new {.next=NULL, .data=#};
while (nondet())
     y = x; x = new {.next=y, .data=?};

while (x->data != #)
    x = x->next; assert(x != NULL);
```

Predicates



Program is safe

END