TLA+/TLC: формальный метод верификации конкурентных алгоритмов для инженеров

Алексей Найдёнов
ITOOLABS

Saint HighLoad++

Amazon Web Services успешно использует TLA+ Tools[1] с 2011 года

> *“At AWS, formal methods have been a big success. They have helped us prevent subtle, serious bugs from reaching production, bugs that we would not have found via any other technique.”*

1. http://lamport.azurewebsites.net/tla/formal-methods-amazon.pdf

Amazon Web Services успешно использует TLA+ Tools[1] с 2011 года

> *“At AWS, formal methods have been a big success. They have helped us prevent subtle, serious bugs from reaching production, bugs that we would not have found via any other technique.”*

Microsoft использует TLA+ во многих проектах[2], включая Azure CosmosDB[3]

> *“TLA+ is not yet a prerequisite for our hiring. However, a candidate’s knowledge of TLA+ is given significant weight in our evaluation. To us, it is a great indicator of those who really care about quality and correctness.”*

1. http://lamport.azurewebsites.net/tla/formal-methods-amazon.pdf
2. https://www.microsoft.com/en-us/research/search/?q=TLA%2B
3. https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels

MongoDB для поиска[1] ошибок в протоколе[2] репликации

> *“We expect that formally modeling our system upfront could have saved 100s of hours of engineering time.”*

1. conf.tlapl.us/07_-_TLAConf19_-_William_Schultz_-_Fixing_a_MongoDB_Replication_Protocol_Bug_with_TLA.pdf
2. github.com/will62794/mongo-repl-tla-models

MongoDB для поиска[1] ошибок в протоколе[2] репликации

> *“We expect that formally modeling our system upfront could have saved 100s of hours of engineering time.”*

CockroachDB для спецификации[3] протокола распределенных транзакций[4]

> *“We found that the process of writing this specification gave us more confidence in the Parallel Commit protocol itself and in its integration into CockroachDB.”*

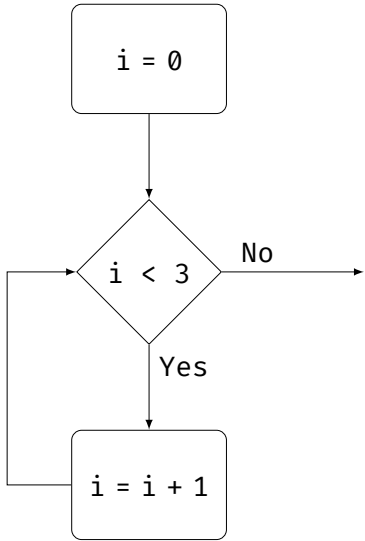
1. conf.tlapl.us/07_-_TLAConf19_-_William_Schultz_-_Fixing_a_MongoDB_Replication_Protocol_Bug_with_TLA.pdf
2. github.com/will62794/mongo-repl-tla-models
3. www.cockroachlabs.com/blog/parallel-commits
4. github.com/cockroachdb/cockroach/tree/master/docs/tla-plus

©The Cartoon Network

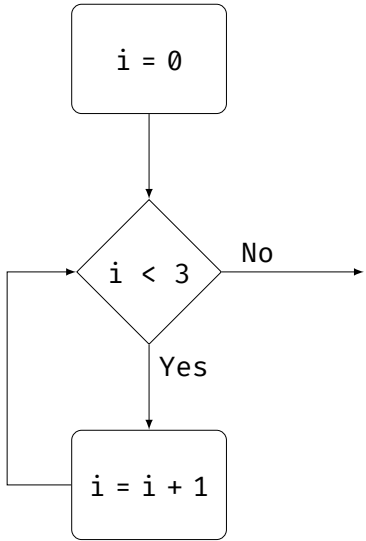
Что такое программа?

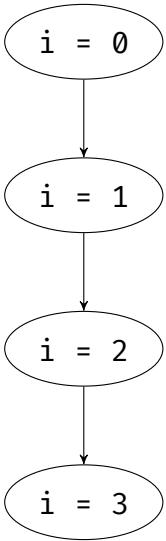
```
var i = 0
for i < 3 {
  i = i + 1
}
```

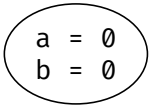
```
var i = 0
for i < 3 {
  i = i + 1
}
```



```
var i = 0
for i < 3 {
  i = i + 1
}
```







Process 1

```
1 a = b + 1
2 b = a + 1
```

Process 2

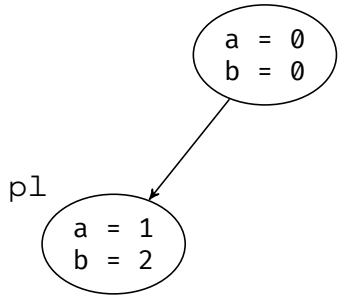
```
1 b = a + 1
2 a = b + 1
```

Process 1

```
1 a = b + 1
2 b = a + 1
```

Process 2

```
1 b = a + 1
2 a = b + 1
```

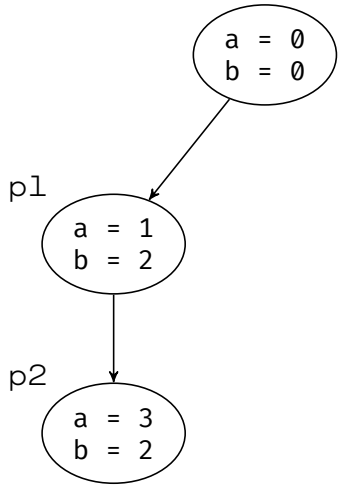


Process 1

```
1 a = b + 1
2 b = a + 1
```

Process 2

```
1 b = a + 1
2 a = b + 1
```

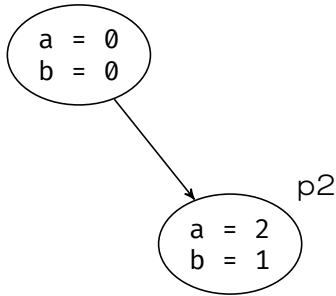


Process 1

```
1 a = b + 1
2 b = a + 1
```

Process 2

```
1 b = a + 1
2 a = b + 1
```

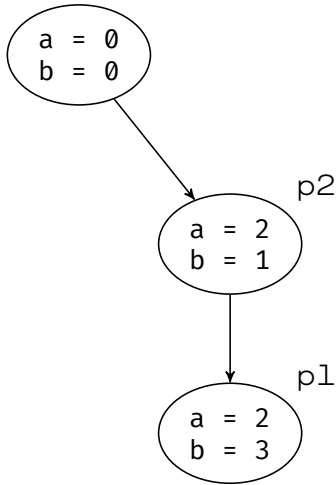


Process 1

```
1 a = b + 1
2 b = a + 1
```

Process 2

```
1 b = a + 1
2 a = b + 1
```

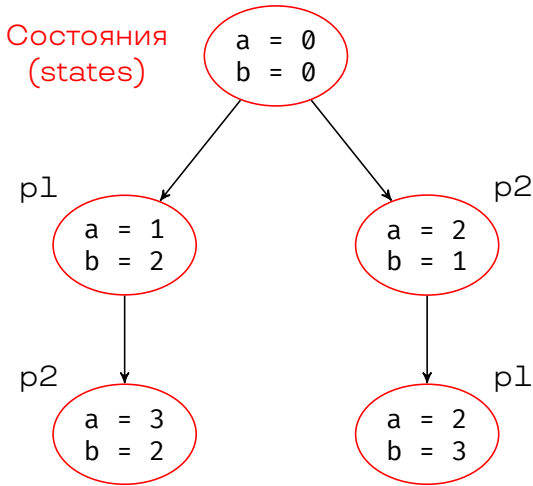


Process 1

```
1 a = b + 1
2 b = a + 1
```

Process 2

```
1 b = a + 1
2 a = b + 1
```

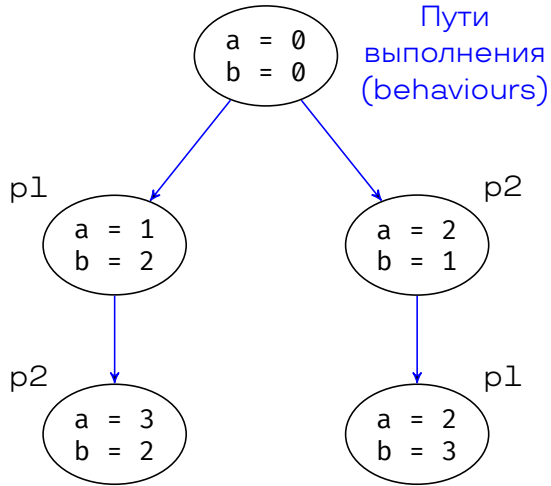


Process 1

```
1 a = b + 1
2 b = a + 1
```

Process 2

```
1 b = a + 1
2 a = b + 1
```

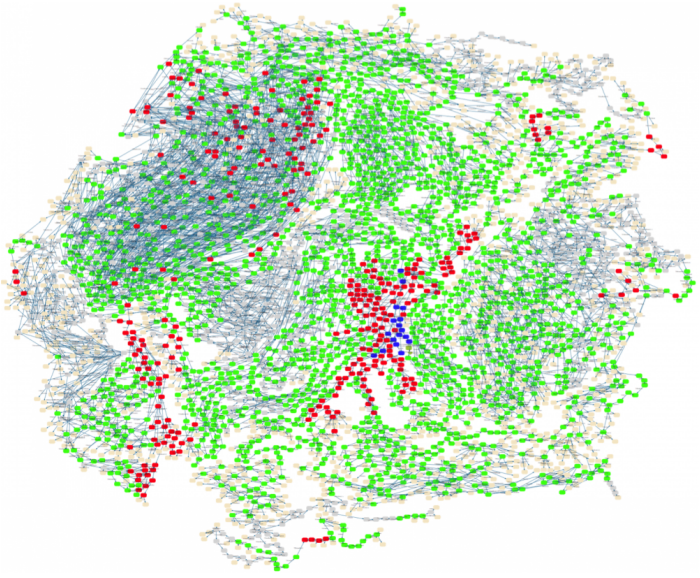


$$(m \cdot n)\frac{(m \cdot n)!}{(n!)^m},$$

где

m — количество процессов

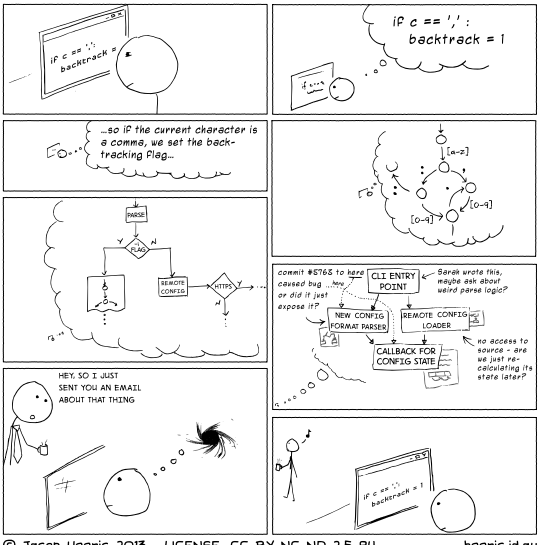
n — длина пути выполнения



CAN Bus state space visualization

https://www3.hhu.de/stups/prob/index.php/State_space_visualization_examples

THIS IS WHY YOU SHOULDN'T INTERRUPT A PROGRAMMER



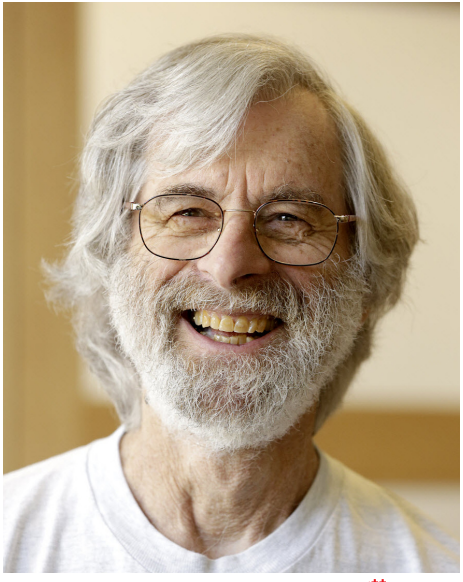
 heeris.id.au

“A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.”

— Leslie Lamport, May 1987

Лесли Лэмпорт (Leslie Lamport)

- Lamport timestamps
- Bakery algorithm
- PAXOS
- LaTeX
- **TLA⁺**



Temporal Logic of Actions

Математическая логика

Математическая логика

$A \wedge B$ Conjunction

Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction

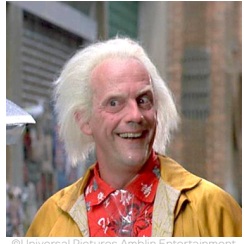
Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction
$\neg A$	Negation

Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction
$\neg A$	Negation

Темпоральная логика



Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction
$\neg A$	Negation

Темпоральная логика

$\Box A$	Always

Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction
$\neg A$	Negation

Темпоральная логика

$\Box A$	Always
$\neg\Box\neg A$	

Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction
$\neg A$	Negation

Темпоральная логика

$\Box A$	Always
$\Diamond A$	Eventually

Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction
$\neg A$	Negation

Темпоральная логика

$\Box A$	Always
$\Diamond A$	Eventually
$\Diamond \Box A$	

Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction
$\neg A$	Negation

Темпоральная логика

$\Box A$	Always
$\Diamond A$	Eventually
$\Diamond \Box A$	Eventually always

Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction
$\neg A$	Negation

Темпоральная логика

$\Box A$	Always
$\Diamond A$	Eventually
$\Diamond \Box A$	Eventually always
p'	

Математическая логика

$A \wedge B$	Conjunction
$A \vee B$	Disjunction
$\neg A$	Negation

Темпоральная логика

$\Box A$	Always
$\Diamond A$	Eventually
$\Diamond \Box A$	Eventually always
p'	Next state

Математическая логика

$A \wedge B$	`A/\B`	Conjunction
$A \vee B$	`A\/B`	Disjunction
$\neg A$	`~A`	Negation

Темпоральная логика

$\Box A$	`[]A`	Always
$\Diamond A$	`<>A`	Eventually
$\Diamond \Box A$	`<>[]A`	Eventually always
p'	`p'`	Next state

TLA+

TRUE TRUE True

TLA+

TRUE	TRUE	True
FALSE	FALSE	False

TLA+

TRUE	TRUE	True
FALSE	FALSE	False
$A \triangleq B$	A == B	Define

TLA+

TRUE	TRUE	True
FALSE	FALSE	False
$A \triangleq B$	A == B	Define
$A = B$	A = B	Equality

TLA+

TRUE	TRUE	True
FALSE	FALSE	False
$A \triangleq B$	A == B	Define
$A = B$	A = B	Equality
$A \neq B$	A /= B	Inequality

TLA+

TRUE	`TRUE`	True
FALSE	`FALSE`	False
$A \triangleq B$	`A == B`	Define
$A = B$	`A = B`	Equality
$A \neq B$	`A /= B`	Inequality
$x \in S$	`x \in S`	x is in S

TLA+

TRUE	`TRUE`	True
FALSE	`FALSE`	False
$A \triangleq B$	`A == B`	Define
$A = B$	`A = B`	Equality
$A \neq B$	`A /= B`	Inequality
$x \in S$	`x \in S`	x is in S
$x \notin S$	`x \notin S`	x is not in S

TLA+

TRUE	`TRUE`	True
FALSE	`FALSE`	False
$A \triangleq B$	`A == B`	Define
$A = B$	`A = B`	Equality
$A \neq B$	`A /= B`	Inequality
$x \in S$	`x \in S`	x is in S
$x \notin S$	`x \notin S`	x is not in S
$\forall x \in S : P$	`\A x \in S: P`	Universal quantifier

TLA+

TRUE	`TRUE`	True
FALSE	`FALSE`	False
$A \triangleq B$	`A == B`	Define
$A = B$	`A = B`	Equality
$A \neq B$	`A /= B`	Inequality
$x \in S$	`x \in S`	x is in S
$x \notin S$	`x \notin S`	x is not in S
$\forall x \in S : P$	`\A x \in S: P`	Universal quantifier
$\exists x \in S : P$	`\E x \in S: P`	Existential quantifier

MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

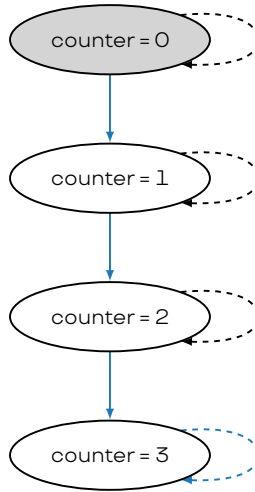
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq \Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq \wedge$ Init
$\wedge \Box[\text{Next}]_{\text{counter}}$
$\wedge \text{WF}_{\text{counter}}(\text{Next})$



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

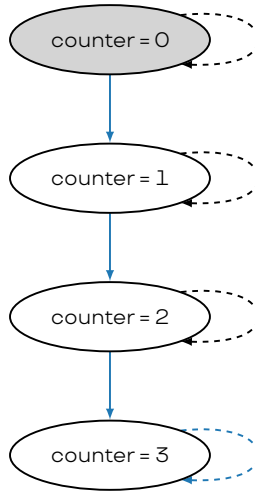
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq$ $\Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq$ $\wedge$ Init
$\wedge$ $\Box$[Next]$_{\text{counter}}$
$\wedge$ WF$_{\text{counter}}$(Next)



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

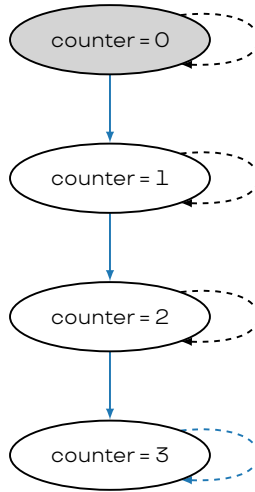
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq$ $\Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq$ $\wedge$ Init
$\wedge$ $\Box$[Next]$_{\text{counter}}$
$\wedge$ WF$_{\text{counter}}$(Next)



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

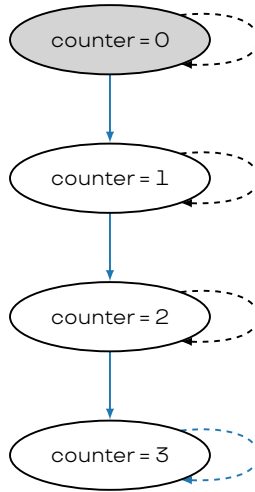
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq$ $\Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq$ $\wedge$ Init
$\wedge$ $\Box$[Next]$_{\text{counter}}$
$\wedge$ WF$_{\text{counter}}$(Next)



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

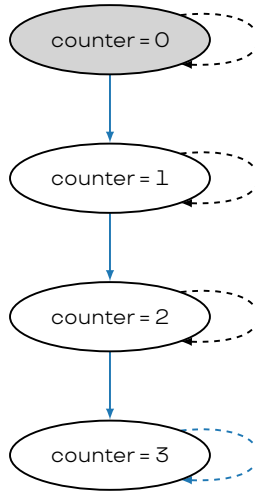
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq \Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq \wedge$ Init
$\wedge \Box[\text{Next}]_{\text{counter}}$
$\wedge \text{WF}_{\text{counter}}(\text{Next})$



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

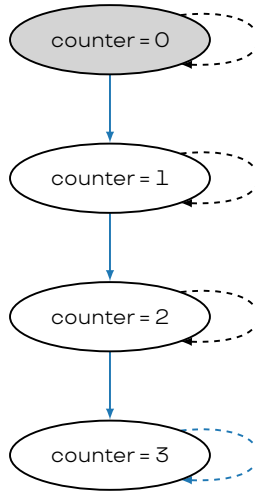
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq \Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq \wedge$ Init
$\wedge \Box[\text{Next}]_{\text{counter}}$
$\wedge \text{WF}_{\text{counter}}(\text{Next})$



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

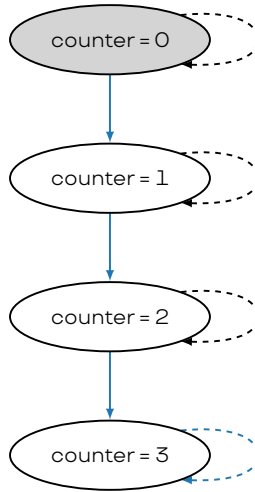
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq \Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq \wedge$ Init
$\wedge \Box[\text{Next}]_{\text{counter}}$
$\wedge \text{WF}_{\text{counter}}(\text{Next})$



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

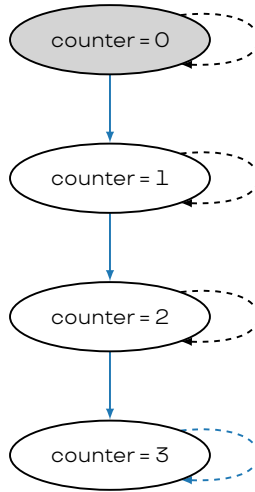
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq$ $\Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq$ $\wedge$ Init
$\wedge$ $\Box$[Next]$_{\text{counter}}$
$\wedge$ WF$_{\text{counter}}$(Next)



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

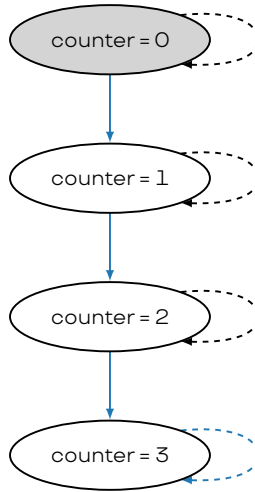
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq \Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq \wedge$ Init
$\wedge \Box[\text{Next}]_{\text{counter}}$
$\wedge \text{WF}_{\text{counter}}(\text{Next})$



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

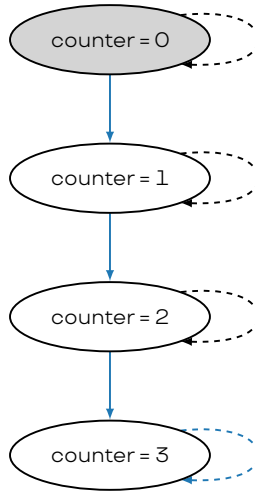
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq$ $\Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq$ $\wedge$ Init
$\wedge$ $\Box$[Next]$_{\text{counter}}$
$\wedge$ WF$_{\text{counter}}$(Next)



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

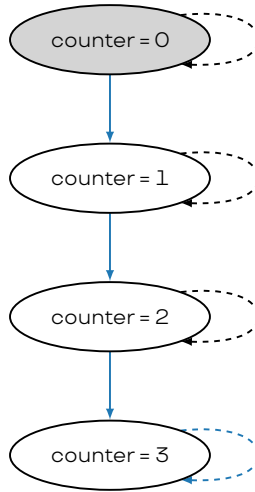
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq \Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

Spec $\triangleq \wedge$ Init
$\wedge \Box[\text{Next}]_{\text{counter}}$
$\wedge \text{WF}_{\text{counter}}(\text{Next})$



MODULE 00_Counter

EXTENDS Naturals
CONSTANTS MinValue, MaxValue
ASSUME MinValue $<$ MaxValue
VARIABLE counter

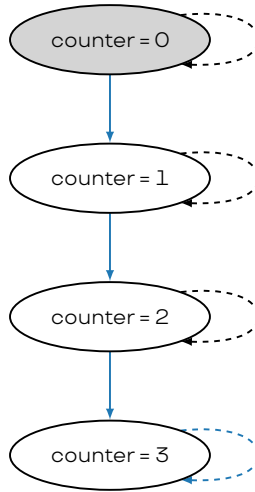
Invariant $\triangleq$ counter $\in$ MinValue..MaxValue

Success $\triangleq$ $\Diamond\Box$(counter $=$ MaxValue)

Init $\triangleq$ counter $=$ MinValue

Next $\triangleq$ counter$'$ $=$ IF counter $<$ MaxValue
THEN counter + 1
ELSE counter

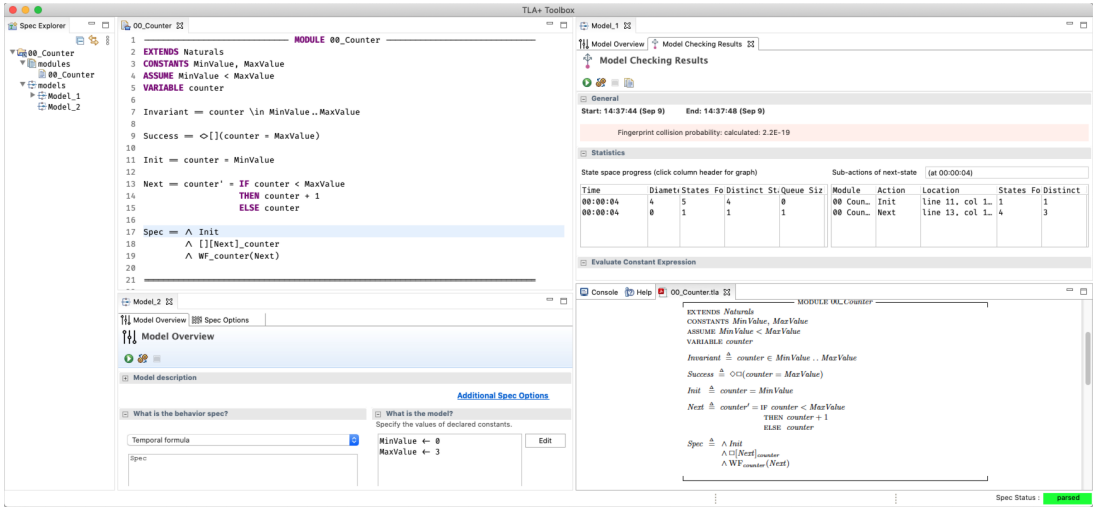
Spec $\triangleq$ $\wedge$ Init
$\wedge$ $\Box[\text{Next}]_{\text{counter}}$
$\wedge$ $\text{WF}_{\text{counter}}(\text{Next})$



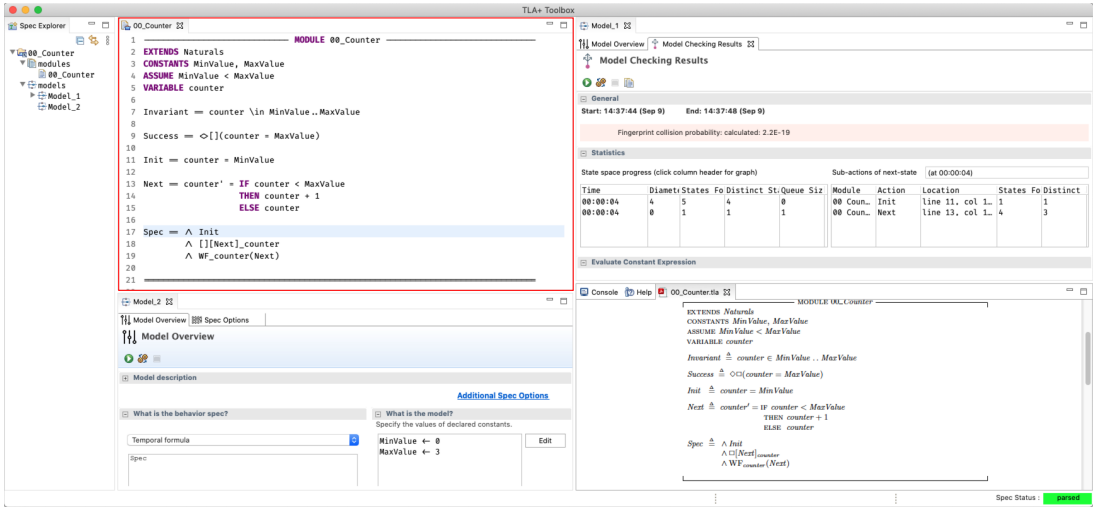
TLC

Explicit State Model Checker for TLA+

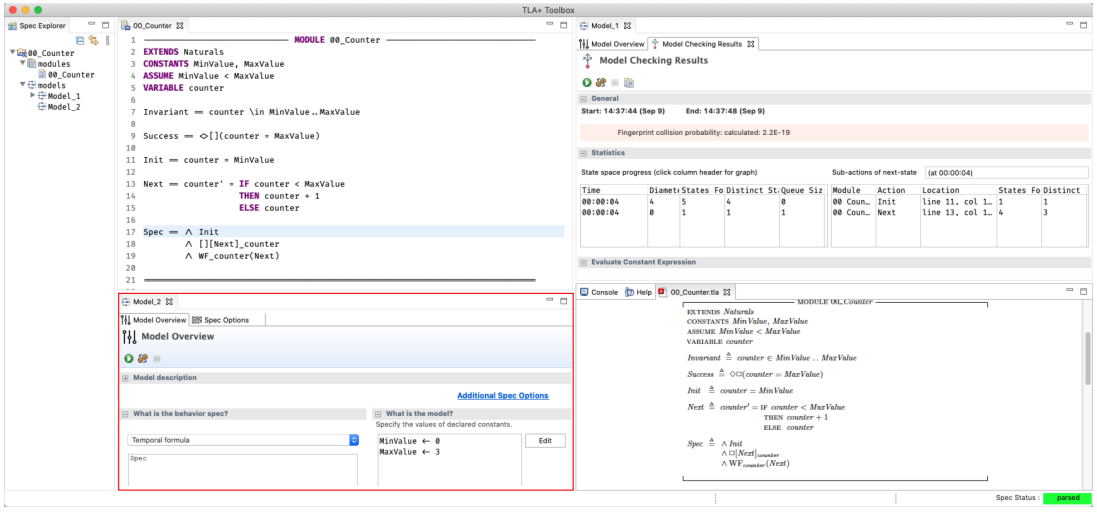
TLA⁺ Toolbox by Simon Zambrovski, et al.



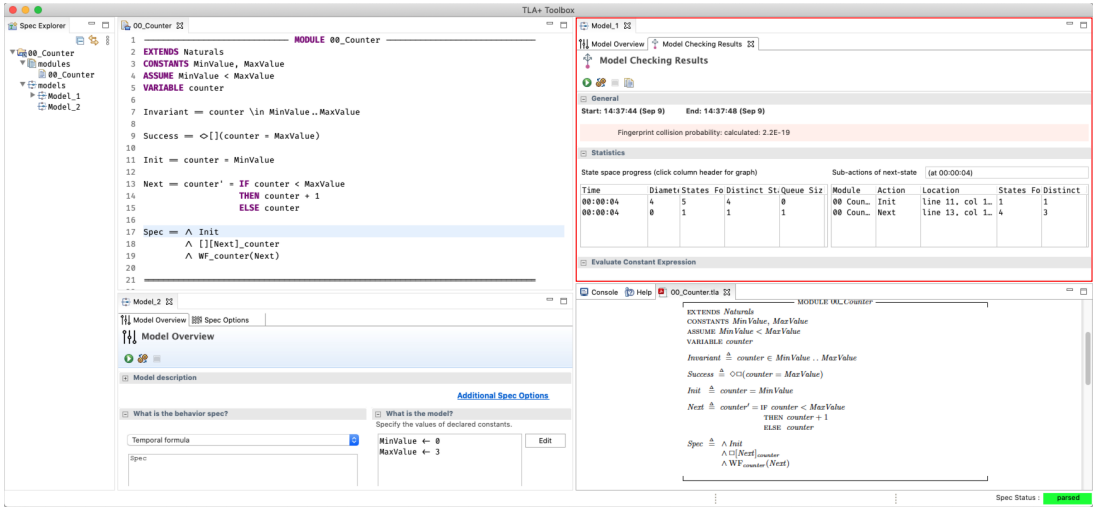
TLA+ Toolbox by Simon Zambrovski, et al.



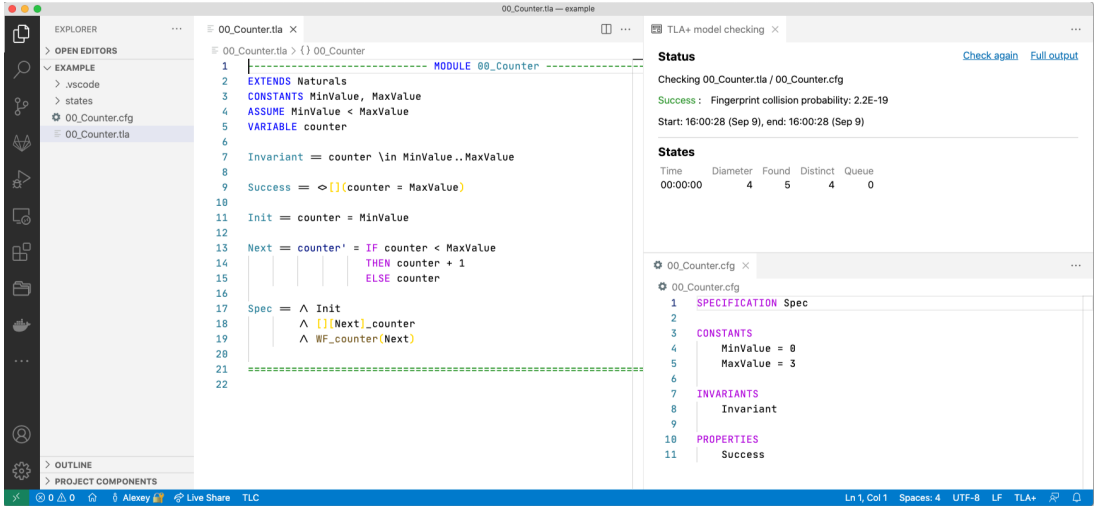
TLA⁺ Toolbox by Simon Zambrovski, et al.



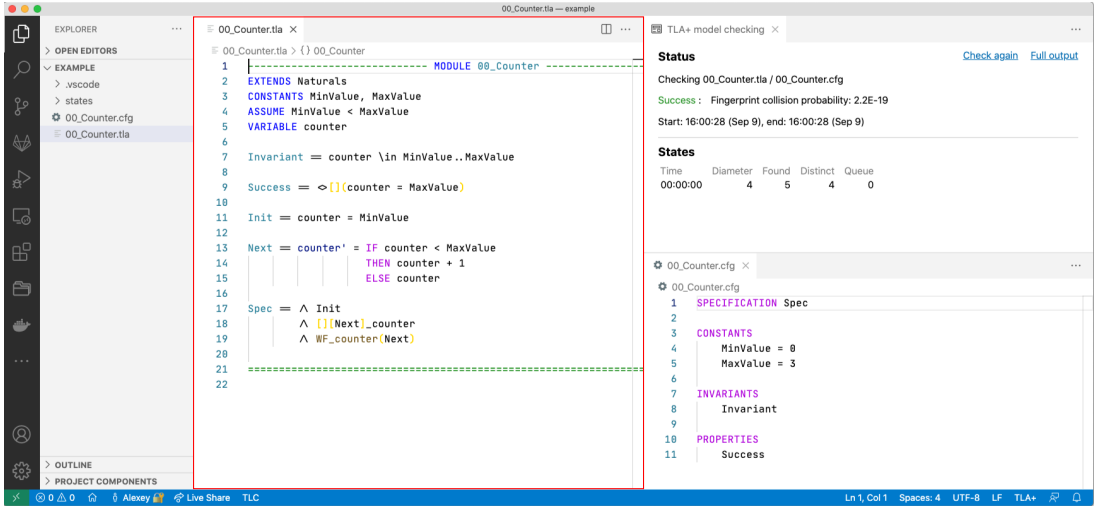
TLA⁺ Toolbox by Simon Zambrovski, et al.



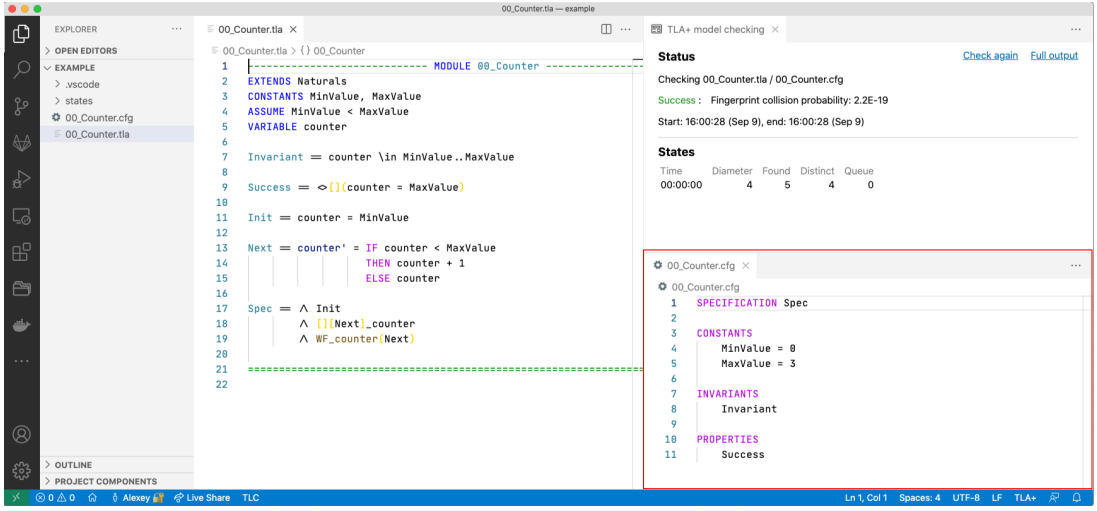
VS Code / TLA+ Support by Андрей Лыгин @alygin



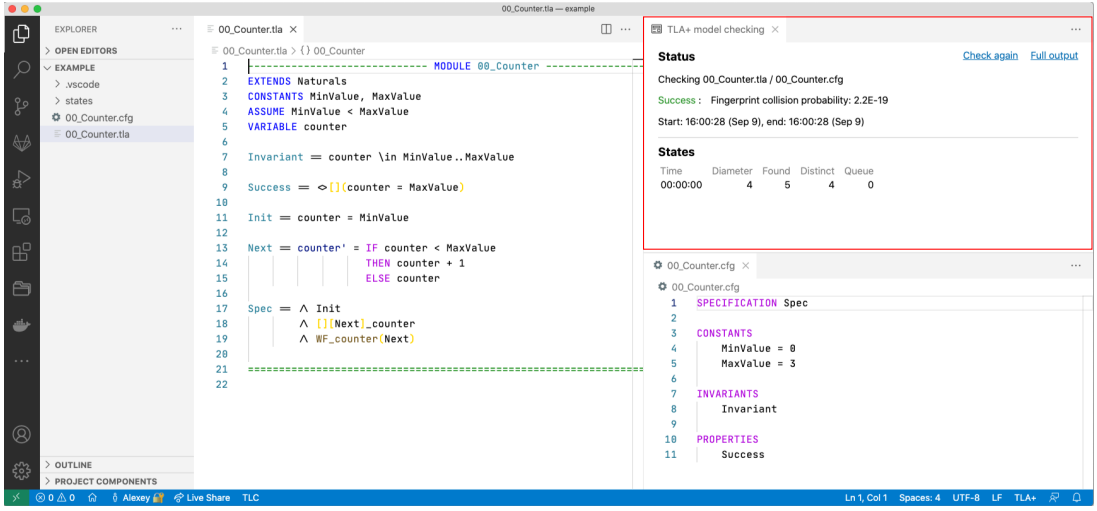
VS Code / TLA⁺ Support by Андрей Лыгин @alygin



VS Code / TLA+ Support by Андрей Лыгин @alygin



VS Code / TLA+ Support by Андрей Лыгин @alygin



PlusCal

PlusCal

```
(*--algorithm 01_PCDemo

variables
  a = 0,
  b = 0;

define
  Success == <>[](a+b = 5)
end define;

fair process proc1 = 1
begin p1:
  a := b + 1;
  b := a + 1;
end process;

fair process proc2 = 2
begin p2:
  b := a + 1;
  a := b + 1;
end process;

end algorithm; *)
```

PlusCal

Алгоритм

```
4  (*--algorithm 01_PCDemo
5
6  variables
7    a = 0,
8    b = 0;
9
10 define
11   Success == <>[](a+b = 5)
12 end define;
```

```
14 fair process proc1 = 1
15 begin p1:
16   a := b + 1;
17   b := a + 1;
18 end process;
19
20 fair process proc2 = 2
21 begin p2:
22   b := a + 1;
23   a := b + 1;
24 end process;
25
26 end algorithm; *)
```

PlusCal

Переменные

```
(*--algorithm 01_PCDemo

variables
  a = 0,
  b = 0;

define
  Success == <>[](a+b = 5)
end define;

fair process proc1 = 1
begin p1:
  a := b + 1;
  b := a + 1;
end process;

fair process proc2 = 2
begin p2:
  b := a + 1;
  a := b + 1;
end process;

end algorithm; *)
```

PlusCal

Определения TLA+

```
4  (*--algorithm 01_PCDemo
5
6  variables
7    a = 0,
8    b = 0;
9
10 define
11   Success == <>[](a+b = 5)
12 end define;
```

```
14 fair process proc1 = 1
15 begin p1:
16   a := b + 1;
17   b := a + 1;
18 end process;
19
20 fair process proc2 = 2
21 begin p2:
22   b := a + 1;
23   a := b + 1;
24 end process;
25
26 end algorithm; *)
```

PlusCal

Процессы

```
(*--algorithm 01_PCDemo

variables
  a = 0,
  b = 0;

define
  Success == <>[](a+b = 5)
end define;
```

```
fair process proc1 = 1
begin p1:
  a := b + 1;
  b := a + 1;
end process;

fair process proc2 = 2
begin p2:
  b := a + 1;
  a := b + 1;
end process;

end algorithm; *)
```

PlusCal

Метки

```
(*--algorithm 01_PCDemo

variables
  a = 0,
  b = 0;

define
  Success == <>[](a+b = 5)
end define;

fair process proc1 = 1
begin p1:
  a := b + 1;
  b := a + 1;
end process;

fair process proc2 = 2
begin p2:
  b := a + 1;
  a := b + 1;
end process;

end algorithm; *)
```

PlusCal

Трансляция

```
27  \* BEGIN TRANSLATION
28  VARIABLES a, b, pc
29
30  (* define statement *)
31  Success == <>[](a+b = 5)
34  vars == << a, b, pc >>
35
36  ProcSet == {1} \cup {2}
37
38  Init == (* Global variables *)
39          /\ a = 0
40          /\ b = 0
41          /\ pc = [self \in ProcSet |-> CASE self = 1 -> "p1"
42                                          [] self = 2 -> "p2"]
```

PlusCal

Трансляция

```
27 \* BEGIN TRANSLATION
28 VARIABLES a, b, pc
29
30 (* define statement *)
31 Success == <>[](a+b = 5)
34 vars == << a, b, pc >>
35
36 ProcSet == {1} \cup {2}
37
38 Init == (* Global variables *)
39         /\ a = 0
40         /\ b = 0
41         /\ pc = [self \in ProcSet |-> CASE self = 1 -> "p1"
42                                        [] self = 2 -> "p2"]
```

PlusCal

Трансляция

```
27 \* BEGIN TRANSLATION
28 VARIABLES a, b, pc
29
30 (* define statement *)
31 Success == <>[](a+b = 5)
34 vars == << a, b, pc >>
35
36 ProcSet == {1} \cup {2}
37
38 Init == (* Global variables *)
39         /\ a = 0
40         /\ b = 0
41         /\ pc = [self \in ProcSet |-> CASE self = 1 -> "p1"
42                                        [] self = 2 -> "p2"]
```

PlusCal

Трансляция

```
44 p1 == /\ pc[1] = "p1"
45       /\ a' = b + 1
46       /\ b' = a' + 1
47       /\ pc' = [pc EXCEPT ![1] = "Done"]
48
49 proc1 == p1
50
51 p2 == /\ pc[2] = "p2"
52       /\ b' = a + 1
53       /\ a' = b' + 1
54       /\ pc' = [pc EXCEPT ![2] = "Done"]
55
56 proc2 == p2
```

PlusCal

Трансляция

```
44  p1 == /\ pc[1] = "p1"
45        /\ a' = b + 1
46        /\ b' = a' + 1
47        /\ pc' = [pc EXCEPT ![1] = "Done"]
48
49  proc1 == p1
50
51  p2 == /\ pc[2] = "p2"
52        /\ b' = a + 1
53        /\ a' = b' + 1
54        /\ pc' = [pc EXCEPT ![2] = "Done"]
55
56  proc2 == p2
```

PlusCal

Трансляция

```
58 (* Allow infinite stuttering to prevent deadlock on termination.
   ↪  *)
59 Terminating == /\ \A self \in ProcSet: pc[self] = "Done"
60                /\ UNCHANGED vars
61
62 Next == proc1 \/ proc2
63           \/ Terminating
64
65 Spec == /\ Init /\ [][Next]_vars
66         /\ WF_vars(proc1)
67         /\ WF_vars(proc2)
68
69 Termination == <>(\A self \in ProcSet: pc[self] = "Done")
```

PlusCal

Трансляция

```
58 (* Allow infinite stuttering to prevent deadlock on termination.
   ↪  *)
59 Terminating == /\ \A self \in ProcSet: pc[self] = "Done"
60                /\ UNCHANGED vars
61
62 Next == proc1 \/ proc2
63           \/ Terminating
64
65 Spec == /\ Init /\ [][Next]_vars
66         /\ WF_vars(proc1)
67         /\ WF_vars(proc2)
68
69 Termination == <>(\A self \in ProcSet: pc[self] = "Done")
```

PlusCal

Трансляция

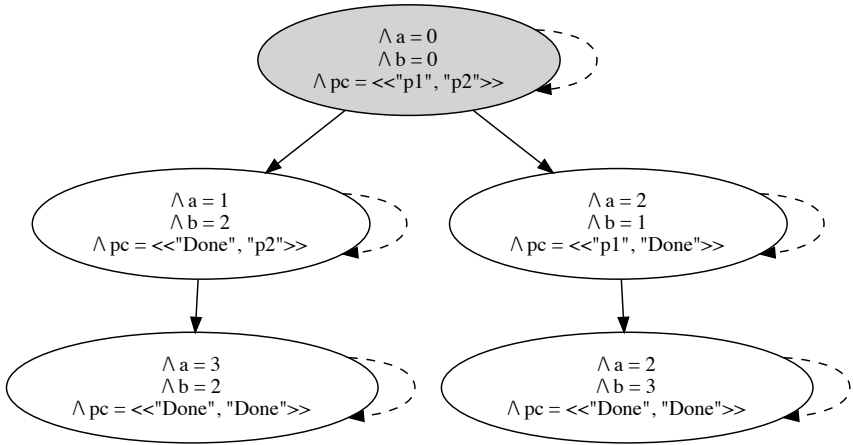
```
58 (* Allow infinite stuttering to prevent deadlock on termination.
   ↪  *)
59 Terminating == /\ \A self \in ProcSet: pc[self] = "Done"
60                /\ UNCHANGED vars
61
62 Next == proc1 \/ proc2
63           \/ Terminating
64
65 Spec == /\ Init /\ [][Next]_vars
66         /\ WF_vars(proc1)
67         /\ WF_vars(proc2)
68
69 Termination == <>(\A self \in ProcSet: pc[self] = "Done")
```

ДЕМО

`01_PCDemo.tla`

TLA+ Toolbox CLI

```
growler@macbook:$ tlc -dump dot,colorized 01_PCDemo.dot 01_PCDemo.tla
Computing initial states...
Finished computing initial states: 1 distinct state generated.
Checking 2 branches of temporal properties for the complete state space
  with 10 total distinct states
Finished checking temporal properties
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
  calculated (optimistic):  val = 5.4E-19
7 states generated, 5 distinct states found, 0 states left on queue.
growler@macbook:$ dot -Tsvg 01_PCDemo.dot -o 01_PCDemo.svg
growler@macbook:$
```



PlusCal

Атомарность

```
4  (*--algorithm A
5
6  variables
7      a = 0,
8      b = 0;
9
10 define
11     Success == <>[](
12         (a + b) = 5
13     )
14 end define;
```

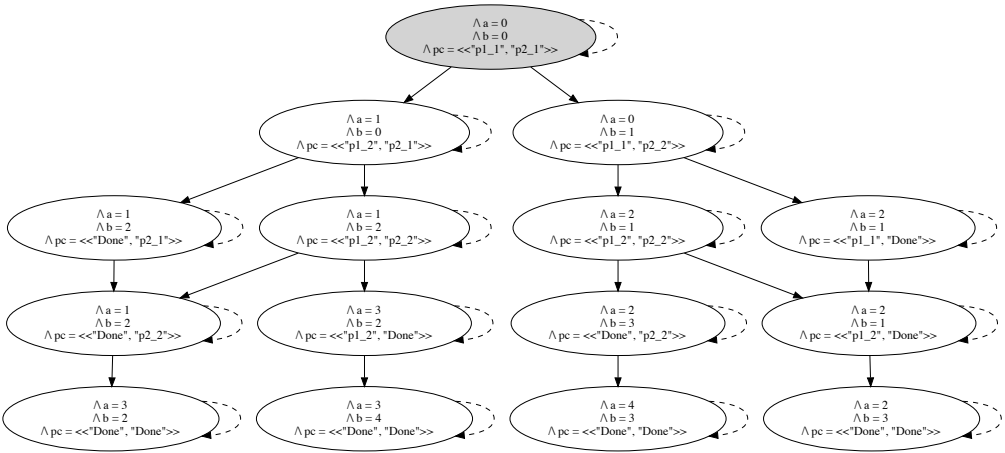
```
16 fair process proc1 = 1
17 begin
18     p1_1: a := b + 1;
19     p1_2: b := a + 1;
20 end process;
21
22 fair process proc2 = 2
23 begin
24     p2_1: b := a + 1;
25     p2_2: a := b + 1;
26 end process;
27
28 end algorithm; *)
```

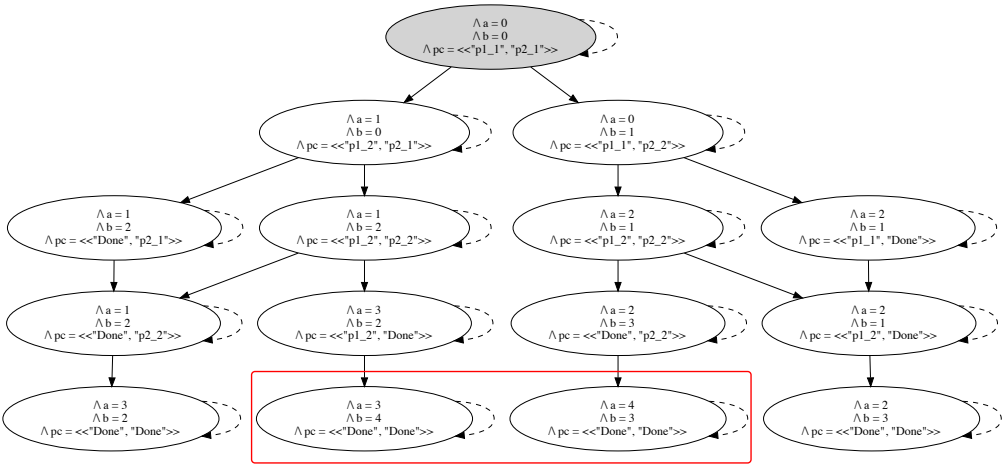
```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 1: <Initial predicate>
/\ a = 0
/\ b = 0
/\ pc = <<"p1_1", "p2_1">>

State 2: <p1_1 line 48, col 9 to line 51, col 17 of module 02_PCDemo>
/\ a = 1
/\ b = 0
/\ pc = <<"p1_2", "p2_1">>

State 3: <p2_1 line 60, col 9 to line 63, col 17 of module 02_PCDemo>
/\ a = 1
/\ b = 2
/\ pc = <<"p1_2", "p2_2">>

State 4: <p2_2 line 65, col 9 to line 68, col 17 of module 02_PCDemo>
/\ a = 3
/\ b = 2
/\ pc = <<"p1_2", "Done">>

State 5: <p1_2 line 53, col 9 to line 56, col 17 of module 02_PCDemo>
/\ a = 3
/\ b = 4
/\ pc = <<"Done", "Done">>
```

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 1: <Initial predicate>
/\ a = 0
/\ b = 0
/\ pc = <<"p1_1", "p2_1">>

State 2: <p1_1 line 48, col 9 to line 51, col 17 of module 02_PCDemo>
/\ a = 1
/\ b = 0
/\ pc = <<"p1_2", "p2_1">>

State 3: <p2_1 line 60, col 9 to line 63, col 17 of module 02_PCDemo>
/\ a = 1
/\ b = 2
/\ pc = <<"p1_2", "p2_2">>

State 4: <p2_2 line 65, col 9 to line 68, col 17 of module 02_PCDemo>
/\ a = 3
/\ b = 2
/\ pc = <<"p1_2", "Done">>

State 5: <p1_2 line 53, col 9 to line 56, col 17 of module 02_PCDemo>
/\ a = 3
/\ b = 4
/\ pc = <<"Done", "Done">>
```

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 1: <Initial predicate>
/\ a = 0
/\ b = 0
/\ pc = <<"p1_1", "p2_1">>

State 2: <p1_1 line 48, col 9 to line 51, col 17 of module 02_PCDemo>
/\ a = 1
/\ b = 0
/\ pc = <<"p1_2", "p2_1">>

State 3: <p2_1 line 60, col 9 to line 63, col 17 of module 02_PCDemo>
/\ a = 1
/\ b = 2
/\ pc = <<"p1_2", "p2_2">>

State 4: <p2_2 line 65, col 9 to line 68, col 17 of module 02_PCDemo>
/\ a = 3
/\ b = 2
/\ pc = <<"p1_2", "Done">>

State 5: <p1_2 line 53, col 9 to line 56, col 17 of module 02_PCDemo>
/\ a = 3
/\ b = 4
/\ pc = <<"Done", "Done">>
```

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 1: <Initial predicate>
/\ a = 0
/\ b = 0
/\ pc = <<"p1_1", "p2_1">>

State 2: <p1_1 line 48, col 9 to line 51, col 17 of module 02_PCDemo>
/\ a = 1
/\ b = 0
/\ pc = <<"p1_2", "p2_1">>

State 3: <p2_1 line 60, col 9 to line 63, col 17 of module 02_PCDemo>
/\ a = 1
/\ b = 2
/\ pc = <<"p1_2", "p2_2">>

State 4: <p2_2 line 65, col 9 to line 68, col 17 of module 02_PCDemo>
/\ a = 3
/\ b = 2
/\ pc = <<"p1_2", "Done">>

State 5: <p1_2 line 53, col 9 to line 56, col 17 of module 02_PCDemo>
/\ a = 3
/\ b = 4
/\ pc = <<"Done", "Done">>
```





TLA+/PlusCal

Sets

```
Procs == {1, 5, 8}
Ops == {"inc", "dec"}
Vals == 1..MaxValue
Cmd == Ops \union {"nop"}
{1, 2, 3} \ {2} = {1, 3}
{1, 2} \intersect {2, 3} = {2}
var \in 1..10
ASSUME Var \notin Procs
1 \in Nat = TRUE
```

TLA+/PlusCal

Sets

```
Procs == {1, 5, 8}
Ops == {"inc", "dec"}
Vals == 1..MaxValue
Cmd == Ops \union {"nop"}
{1, 2, 3} \ {2} = {1, 3}
{1, 2} \intersect {2, 3} = {2}
var \in 1..10
ASSUME Var \notin Procs
1 \in Nat = TRUE
```

TLA+/PlusCal

Sets

```
Procs == {1, 5, 8}
Ops == {"inc", "dec"}
Vals == 1..MaxValue
Cmd == Ops \union {"nop"}
{1, 2, 3} \ {2} = {1, 3}
{1, 2} \intersect {2, 3} = {2}
var \in 1..10
ASSUME Var \notin Procs
1 \in Nat = TRUE
```

TLA+/PlusCal

Sets

```
Procs == {1, 5, 8}
Ops == {"inc", "dec"}
Vals == 1..MaxValue
Cmd == Ops \union {"nop"}
{1, 2, 3} \ {2} = {1, 3}
{1, 2} \intersect {2, 3} = {2}
var \in 1..10
ASSUME Var \notin Procs
1 \in Nat = TRUE
```

TLA+/PlusCal

Sets

```
Procs == {1, 5, 8}
Ops == {"inc", "dec"}
Vals == 1..MaxValue
Cmd == Ops \union {"nop"}
{1, 2, 3} \ {2} = {1, 3}
{1, 2} \intersect {2, 3} = {2}
var \in 1..10
ASSUME Var \notin Procs
1 \in Nat = TRUE
```

TLA+/PlusCal

Sets

```
Procs == {1, 5, 8}
Ops == {"inc", "dec"}
Vals == 1..MaxValue
Cmd == Ops \union {"nop"}
{1, 2, 3} \ {2} = {1, 3}
{1, 2} \intersect {2, 3} = {2}
var \in 1..10
ASSUME Var \notin Procs
1 \in Nat = TRUE
```

TLA+/PlusCal

Sets

```
Procs == {1, 5, 8}
Ops == {"inc", "dec"}
Vals == 1..MaxValue
Cmd == Ops \union {"nop"}
{1, 2, 3} \ {2} = {1, 3}
{1, 2} \intersect {2, 3} = {2}
var \in 1..10
ASSUME Var \notin Procs
1 \in Nat = TRUE
```

TLA+/PlusCal

Sets

```
Procs == {1, 5, 8}
Ops == {"inc", "dec"}
Vals == 1..MaxValue
Cmd == Ops \union {"nop"}
{1, 2, 3} \ {2} = {1, 3}
{1, 2} \intersect {2, 3} = {2}
var \in 1..10
ASSUME Var \notin Procs
1 \in Nat = TRUE
```

TLA+/PlusCal

Functions

```
Squares == [n \in Nat |-> n * n]
Squares[2] = 4
DOMAIN Squares = Nat
F == [n \in 1..3 |-> n * n]
DOMAIN F = 1..3
Q == [F EXCEPT ![2] = 0]
{Q[i]: i \in DOMAIN Q} = {1, 0, 9}
```

TLA+/PlusCal

Functions

```
1 Squares == [n \in Nat |-> n * n]
2 Squares[2] = 4
3 DOMAIN Squares = Nat
4 F == [n \in 1..3 |-> n * n]
5 DOMAIN F = 1..3
6 Q == [F EXCEPT ![2] = 0]
7 {Q[i]: i \in DOMAIN Q} = {1, 0, 9}
```

TLA+/PlusCal

Functions

```
1 Squares == [n \in Nat |-> n * n]
2 Squares[2] = 4
3 DOMAIN Squares = Nat
4 F == [n \in 1..3 |-> n * n]
5 DOMAIN F = 1..3
6 Q == [F EXCEPT ![2] = 0]
7 {Q[i]: i \in DOMAIN Q} = {1, 0, 9}
```

TLA+/PlusCal

Functions

```
Squares == [n \in Nat |-> n * n]
Squares[2] = 4
DOMAIN Squares = Nat
F == [n \in 1..3 |-> n * n]
DOMAIN F = 1..3
Q == [F EXCEPT ![2] = 0]
{Q[i]: i \in DOMAIN Q} = {1, 0, 9}
```

TLA+/PlusCal

Functions

```
1 Squares == [n \in Nat |-> n * n]
2 Squares[2] = 4
3 DOMAIN Squares = Nat
4 F == [n \in 1..3 |-> n * n]
5 DOMAIN F = 1..3
6 Q == [F EXCEPT ![2] = 0]
7 {Q[i]: i \in DOMAIN Q} = {1, 0, 9}
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Tuples

```
a == <<1, 5, 8>>
DOMAIN a = 1..3
a[2] = 5
Len(a) = 3
Head(a) = 1
Tail(a) = <<5, 8>>
SubSeq(a, 1, 2) = <<1, 5>>
SelectSeq(a, LAMBDA i: i > 5) = <<8>>
<<1, 2>> \o <<4, 8>> = <<1, 2, 4, 8>>
a := Append(a, 5)
a[i] := i
```

TLA+/PlusCal

Records

```
s == [rdy|->0, ack|->0, val|->0]
DOMAIN s = {"rdy", "ack", "val"}
s.rdy = 0
s["ack"] = 0
"a" :> 1 = [a|->1]
[a->1] @@ [b|->2] = [a|->1, b|->2]
[s EXCEPT !.rdy = @ + 1] = [rdy->1, ack|->0, val|->0]
s.ack := 1
[i \in DOMAIN s \ {"ack"} |-> s[i]] = [rdy|->0, val|->0]
```

TLA+/PlusCal

Records

```
s == [rdy|->0, ack|->0, val|->0]
DOMAIN s = {"rdy", "ack", "val"}
s.rdy = 0
s["ack"] = 0
"a" :> 1 = [a|->1]
[a->1] @@ [b|->2] = [a|->1, b|->2]
[s EXCEPT !.rdy = @ + 1] = [rdy->1, ack|->0, val|->0]
s.ack := 1
[i \in DOMAIN s \ {"ack"} |-> s[i]] = [rdy|->0, val|->0]
```

TLA+/PlusCal

Records

```
s == [rdy|->0, ack|->0, val|->0]
DOMAIN s = {"rdy", "ack", "val"}
s.rdy = 0
s["ack"] = 0
"a" :> 1 = [a|->1]
[a->1] @@ [b|->2] = [a|->1, b|->2]
[s EXCEPT !.rdy = @ + 1] = [rdy->1, ack|->0, val|->0]
s.ack := 1
[i \in DOMAIN s \ {"ack"} |-> s[i]] = [rdy|->0, val|->0]
```

TLA+/PlusCal

Records

```
s == [rdy|->0, ack|->0, val|->0]
DOMAIN s = {"rdy", "ack", "val"}
s.rdy = 0
s["ack"] = 0
"a" :> 1 = [a|->1]
[a->1] @@ [b|->2] = [a|->1, b|->2]
[s EXCEPT !.rdy = @ + 1] = [rdy->1, ack|->0, val|->0]
s.ack := 1
[i \in DOMAIN s \ {"ack"} |-> s[i]] = [rdy|->0, val|->0]
```

TLA+/PlusCal

Records

```
s == [rdy|->0, ack|->0, val|->0]
DOMAIN s = {"rdy", "ack", "val"}
s.rdy = 0
s["ack"] = 0
"a" :> 1 = [a|->1]
[a->1] @@ [b|->2] = [a|->1, b|->2]
[s EXCEPT !.rdy = @ + 1] = [rdy->1, ack|->0, val|->0]
s.ack := 1
[i \in DOMAIN s \ {"ack"} |-> s[i]] = [rdy|->0, val|->0]
```

TLA+/PlusCal

Records

```
s == [rdy|->0, ack|->0, val|->0]
DOMAIN s = {"rdy", "ack", "val"}
s.rdy = 0
s["ack"] = 0
"a" :> 1 = [a|->1]
[a->1] @@ [b|->2] = [a|->1, b|->2]
[s EXCEPT !.rdy = @ + 1] = [rdy->1, ack|->0, val|->0]
s.ack := 1
[i \in DOMAIN s \ {"ack"} |-> s[i]] = [rdy|->0, val|->0]
```

TLA+/PlusCal

Records

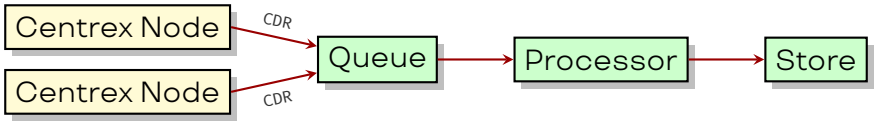
```
s == [rdy|->0, ack|->0, val|->0]
DOMAIN s = {"rdy", "ack", "val"}
s.rdy = 0
s["ack"] = 0
"a" :> 1 = [a|->1]
[a->1] @@ [b|->2] = [a|->1, b|->2]
[s EXCEPT !.rdy = @ + 1] = [rdy->1, ack|->0, val|->0]
s.ack := 1
[i \in DOMAIN s \ {"ack"} |-> s[i]] = [rdy|->0, val|->0]
```

TLA+/PlusCal

Records

```
s == [rdy|->0, ack|->0, val|->0]
DOMAIN s = {"rdy", "ack", "val"}
s.rdy = 0
s["ack"] = 0
"a" :> 1 = [a|->1]
[a->1] @@ [b|->2] = [a|->1, b|->2]
[s EXCEPT !.rdy = @ + 1] = [rdy->1, ack|->0, val|->0]
s.ack := 1
[i \in DOMAIN s \ {"ack"} |-> s[i]] = [rdy|->0, val|->0]
```

Задача



Модель:
требования

CallHistory.tla v0

```
------ MODULE CallHistory ------

EXTENDS Sequences, Integers, TLC

CONSTANTS Calls

(*--algorithm CallHistory

variables
    queue = <<>>,
    store = <<>>;
define

    AllCallsProcessed == <>[](
        \A i \in Calls:
           Len(SelectSeq(store, LAMBDA
            ↪  j: j = i)) = 1
    )

    Success == AllCallsProcessed

end define;

fair process Call \in Calls
begin CallSetup:
    skip;
end process;

fair process Processor = "Processor"
begin Process:
    while TRUE do
ProcessCDR:
        skip;
    end while;
end process;

end algorithm; *)
```

CallHistory.tla v0

```
------ MODULE CallHistory ------

EXTENDS Sequences, Integers, TLC

CONSTANTS Calls

(*--algorithm CallHistory

variables
    queue = <<>>,
    store = <<>>;
define

    AllCallsProcessed == <>[](
        \A i \in Calls:
           Len(SelectSeq(store, LAMBDA
            ↪  j: j = i)) = 1
    )

    Success == AllCallsProcessed

end define;

fair process Call \in Calls
begin CallSetup:
    skip;
end process;

fair process Processor = "Processor"
begin Process:
    while TRUE do
ProcessCDR:
        skip;
    end while;
end process;

end algorithm; *)
```

CallHistory.tla v0

```
------ MODULE CallHistory ------

EXTENDS Sequences, Integers, TLC

CONSTANTS Calls

(*--algorithm CallHistory

variables
    queue = <<>>,
    store = <<>>;
define

    AllCallsProcessed == <>[](
        \A i \in Calls:
           Len(SelectSeq(store, LAMBDA
            ↪  j: j = i)) = 1
    )

    Success == AllCallsProcessed

end define;

fair process Call \in Calls
begin CallSetup:
    skip;
end process;

fair process Processor = "Processor"
begin Process:
    while TRUE do
ProcessCDR:
        skip;
    end while;
end process;

end algorithm; *)
```

CallHistory.tla v0

```
------ MODULE CallHistory ------

EXTENDS Sequences, Integers, TLC

CONSTANTS Calls

(*--algorithm CallHistory

variables
    queue = <<>>,
    store = <<>>;
define

    AllCallsProcessed == <>[](
        \A i \in Calls:
           Len(SelectSeq(store, LAMBDA
            ↪  j: j = i)) = 1
    )

    Success == AllCallsProcessed

end define;

fair process Call \in Calls
begin CallSetup:
    skip;
end process;

fair process Processor = "Processor"
begin Process:
    while TRUE do
ProcessCDR:
        skip;
    end while;
end process;

end algorithm; *)
```

CallHistory.tla v0

```
------ MODULE CallHistory ------

EXTENDS Sequences, Integers, TLC

CONSTANTS Calls

(*--algorithm CallHistory

variables
    queue = <<>>,
    store = <<>>;
define

    AllCallsProcessed == <>[](
        \A i \in Calls:
           Len(SelectSeq(store, LAMBDA
            ↪  j: j = i)) = 1
    )

    Success == AllCallsProcessed

end define;

fair process Call \in Calls
begin CallSetup:
    skip;
end process;

fair process Processor = "Processor"
begin Process:
    while TRUE do
ProcessCDR:
        skip;
    end while;
end process;

end algorithm; *)
```

CallHistory.tla v0

```
------ MODULE CallHistory ------

EXTENDS Sequences, Integers, TLC

CONSTANTS Calls

(*--algorithm CallHistory

variables
    queue = <<>>,
    store = <<>>;
define

    AllCallsProcessed == <>[](
        \A i \in Calls:
           Len(SelectSeq(store, LAMBDA
            ↪  j: j = i)) = 1
    )

    Success == AllCallsProcessed

end define;

fair process Call \in Calls
begin CallSetup:
    skip;
end process;

fair process Processor = "Processor"
begin Process:
    while TRUE do
ProcessCDR:
        skip;
    end while;
end process;

end algorithm; *)
```

CallHistory.tla v0

```
------ MODULE CallHistory ------

EXTENDS Sequences, Integers, TLC

CONSTANTS Calls

(*--algorithm CallHistory

variables
    queue = <<>>,
    store = <<>>;
define

    AllCallsProcessed == <>[](
        \A i \in Calls:
           Len(SelectSeq(store, LAMBDA
            ↪  j: j = i)) = 1
    )

    Success == AllCallsProcessed

end define;

fair process Call \in Calls
begin CallSetup:
    skip;
end process;

fair process Processor = "Processor"
begin Process:
    while TRUE do
ProcessCDR:
        skip;
    end while;
end process;

end algorithm; *)
```

CallHistory.cfg v0

```
SPECIFICATION Spec
\* Add statements after this line.

CONSTANTS
    Calls = {"call-1", "call-2", "call-3"}

PROPERTIES
    Success
```

CallHistory.cfg v0

```
SPECIFICATION Spec
\* Add statements after this line.

CONSTANTS
    Calls = {"call-1", "call-2", "call-3"}

PROPERTIES
    Success
```

CallHistory.tla v0 Run: Failed

```
19 Progress(5) at 2021-09-15 20:52:53: 41 states generated, 16 distinct states found, 0
   ↪ states left on queue.
20 Checking temporal properties for the complete state space with 16 total distinct states
   ↪ at (2021-09-15 20:52:53)
21 Error: Temporal properties were violated.
22
23 Error: The following behavior constitutes a counter-example:
64
65 State 6: <ProcessCDR line 70, col 15 to line 73, col 45 of module CallHistory>
66 /\ store = <<>>
67 /\ pc = ( "call-1" :> "Done" @@
68   "call-2" :> "Done" @@
69   "call-3" :> "Done" @@
70   "Processor" :> "Process" )
71 /\ queue = <<>>
72
73 Back to state 5: <Process line 66, col 12 to line 68, col 42 of module CallHistory>
74
75 Finished checking temporal properties in 00s at 2021-09-15 20:52:53
76 41 states generated, 16 distinct states found, 0 states left on queue.
```

CallHistory.tla v0 Run: Failed

```
19 Progress(5) at 2021-09-15 20:52:53: 41 states generated, 16 distinct states found, 0
   ↪ states left on queue.
20 Checking temporal properties for the complete state space with 16 total distinct states
   ↪ at (2021-09-15 20:52:53)
21 Error: Temporal properties were violated.
22
23 Error: The following behavior constitutes a counter-example:
64
65 State 6: <ProcessCDR line 70, col 15 to line 73, col 45 of module CallHistory>
66 /\ store = <<>>
67 /\ pc = ( "call-1" :> "Done" @@
68   "call-2" :> "Done" @@
69   "call-3" :> "Done" @@
70   "Processor" :> "Process" )
71 /\ queue = <<>>
72
73 Back to state 5: <Process line 66, col 12 to line 68, col 42 of module CallHistory>
74
75 Finished checking temporal properties in 00s at 2021-09-15 20:52:53
76 41 states generated, 16 distinct states found, 0 states left on queue.
```

CallHistory.tla v0 Run: Failed

```
19 Progress(5) at 2021-09-15 20:52:53: 41 states generated, 16 distinct states found, 0
   ↪ states left on queue.
20 Checking temporal properties for the complete state space with 16 total distinct states
   ↪ at (2021-09-15 20:52:53)
21 Error: Temporal properties were violated.
22
23 Error: The following behavior constitutes a counter-example:
64
65 State 6: <ProcessCDR line 70, col 15 to line 73, col 45 of module CallHistory>
66 /\ store = <<>>
67 /\ pc = ( "call-1" :> "Done" @@
68   "call-2" :> "Done" @@
69   "call-3" :> "Done" @@
70   "Processor" :> "Process" )
71 /\ queue = <<>>
72
73 Back to state 5: <Process line 66, col 12 to line 68, col 42 of module CallHistory>
74
75 Finished checking temporal properties in 00s at 2021-09-15 20:52:53
76 41 states generated, 16 distinct states found, 0 states left on queue.
```

CallHistory.tla v0 Run: Failed

```
19 Progress(5) at 2021-09-15 20:52:53: 41 states generated, 16 distinct states found, 0
   ↪ states left on queue.
20 Checking temporal properties for the complete state space with 16 total distinct states
   ↪ at (2021-09-15 20:52:53)
21 Error: Temporal properties were violated.
22
23 Error: The following behavior constitutes a counter-example:
64
65 State 6: <ProcessCDR line 70, col 15 to line 73, col 45 of module CallHistory>
66 /\ store = <<>>
67 /\ pc = ( "call-1" :> "Done" @@
68   "call-2" :> "Done" @@
69   "call-3" :> "Done" @@
70   "Processor" :> "Process" )
71 /\ queue = <<>>
72
73 Back to state 5: <Process line 66, col 12 to line 68, col 42 of module CallHistory>
74
75 Finished checking temporal properties in 00s at 2021-09-15 20:52:53
76 41 states generated, 16 distinct states found, 0 states left on queue.
```

CallHistory.tla v0 Run: Failed

```
19 Progress(5) at 2021-09-15 20:52:53: 41 states generated, 16 distinct states found, 0
   ↪ states left on queue.
20 Checking temporal properties for the complete state space with 16 total distinct states
   ↪ at (2021-09-15 20:52:53)
21 Error: Temporal properties were violated.
22
23 Error: The following behavior constitutes a counter-example:
64
65 State 6: <ProcessCDR line 70, col 15 to line 73, col 45 of module CallHistory>
66 /\ store = <<>>
67 /\ pc = ( "call-1" :> "Done" @@
68   "call-2" :> "Done" @@
69   "call-3" :> "Done" @@
70   "Processor" :> "Process" )
71 /\ queue = <<>>
72
73 Back to state 5: <Process line 66, col 12 to line 68, col 42 of module CallHistory>
74
75 Finished checking temporal properties in 00s at 2021-09-15 20:52:53
76 41 states generated, 16 distinct states found, 0 states left on queue.
```

Модель:
первая реализация

CallHistory.tla v0->v1

```
15  (*--algorithm CallHistory
16
32  fair process Call \in Calls
33  begin CallSetup:
34  -     skip;
35  +     queue := Append(queue, [id |-> self, action |-> "setup"]);
36  +CallComplete:
37  +     queue := Append(queue, [id |-> self, action |-> "complete"]);
38  end process;
39
```

CallHistory.tla v0->v1

```
15  (*--algorithm CallHistory
16
32  fair process Call \in Calls
33  begin CallSetup:
34-     skip;
35+     queue := Append(queue, [id |-> self, action |-> "setup"]);
36+CallComplete:
37+     queue := Append(queue, [id |-> self, action |-> "complete"]);
38  end process;
39
```

CallHistory.tla v0->v1

```
 CONSTANTS Calls

+Add(dict, key, val) == dict @@ (key :> val)
+Remove(dict, key) == [i \in DOMAIN dict \ {key} |-> dict[i]]
+
 (*--algorithm CallHistory

 variables
     queue = <<>>,
+    commited = 0,
     store = <<>>;
 define

     AllCallsProcessed == <>[](
         \A i \in Calls:
            Len(SelectSeq(store, LAMBDA j: j = i)) = 1
     )

     Success == AllCallsProcessed

 end define;
```

CallHistory.tla v0->v1

```
 CONSTANTS Calls

+Add(dict, key, val) == dict @@ (key :> val)
+Remove(dict, key) == [i \in DOMAIN dict \ {key} |-> dict[i]]
+
 (*--algorithm CallHistory

 variables
     queue = <<>>,
+    commited = 0,
     store = <<>>;
 define

     AllCallsProcessed == <>[](
         \A i \in Calls:
            Len(SelectSeq(store, LAMBDA j: j = i)) = 1
     )

     Success == AllCallsProcessed

 end define;
```

CallHistory.tla v0->v1

```
 CONSTANTS Calls

+Add(dict, key, val) == dict @@ (key :> val)
+Remove(dict, key) == [i \in DOMAIN dict \ {key} |-> dict[i]]
+
 (*--algorithm CallHistory

 variables
     queue = <<>>,
+    commited = 0,
     store = <<>>;
 define

     AllCallsProcessed == <>[](
         \A i \in Calls:
            Len(SelectSeq(store, LAMBDA j: j = i)) = 1
     )

     Success == AllCallsProcessed

 end define;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v0->v1

```
 fair process Processor = "Processor"
+variables
+    wip = <<>>,
+    offset = commited + 1;
 begin Process:
     while TRUE do
+        when Len(queue) >= offset;
 ProcessCDR:
-        skip;
+        with cdr = queue[offset]
+        do
+            if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
+                wip := Add(wip, cdr.id, [offset |-> offset]);
+            elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
+                wip := Remove(wip, cdr.id);
+                store := Append(store, cdr.id);
+            end if;
+        end with;
+        commited := offset;
+        offset := offset + 1;
     end while;
 end process;
```

CallHistory.tla v1 Run: Succeed

```
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
  calculated (optimistic):  val = 3.2E-13
4927 states generated, 2869 distinct states found, 0 states left on queue.
The depth of the complete state graph search is 19.
The average outdegree of the complete state graph is 1 (minimum is 0, the maximum 4 and
 ↪  the 95th percentile is 3).
```

CallHistory.tla v1 Run: Succeed

```
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
  calculated (optimistic):  val = 3.2E-13
4927 states generated, 2869 distinct states found, 0 states left on queue.
The depth of the complete state graph search is 19.
The average outdegree of the complete state graph is 1 (minimum is 0, the maximum 4 and
 ↪  the 95th percentile is 3).
```

Модель:
добавим graceful shutdown

CallHistory.tla v1->v2

```
15  (*--algorithm CallHistory
16
17  variables
18      queue = <<>>,
19      commited = 0,
20  +   shutdownProcessor = FALSE,
21      store = <<>>;
32
33  fair process Call \in Calls
34  begin CallSetup:
35      queue := Append(queue, [id |-> self, action |-> "setup"]);
36  CallComplete:
37      queue := Append(queue, [id |-> self, action |-> "complete"]);
38  end process;
39
40  +fair process Shutdown = "Shutdown"
41  +begin SignalShutdown:
42  +   when \A c \in Calls: pc[c] = "Done";
43  +   shutdownProcessor := TRUE;
44  +end process;
```

CallHistory.tla v1->v2

```
15  (*--algorithm CallHistory
16
17  variables
18      queue = <<>>,
19      commited = 0,
20+     shutdownProcessor = FALSE,
21      store = <<>>;
32
33  fair process Call \in Calls
34  begin CallSetup:
35      queue := Append(queue, [id |-> self, action |-> "setup"]);
36  CallComplete:
37      queue := Append(queue, [id |-> self, action |-> "complete"]);
38  end process;
39
40 +fair process Shutdown = "Shutdown"
41 +begin SignalShutdown:
42 +     when \A c \in Calls: pc[c] = "Done";
43 +     shutdownProcessor := TRUE;
44 +end process;
```

CallHistory.tla v1->v2

```
15  (*--algorithm CallHistory
16
17  variables
18      queue = <<>>,
19      commited = 0,
20  +   shutdownProcessor = FALSE,
21      store = <<>>;
32
33  fair process Call \in Calls
34  begin CallSetup:
35      queue := Append(queue, [id |-> self, action |-> "setup"]);
36  CallComplete:
37      queue := Append(queue, [id |-> self, action |-> "complete"]);
38  end process;
39
40  +fair process Shutdown = "Shutdown"
41  +begin SignalShutdown:
42  +   when \A c \in Calls: pc[c] = "Done";
43  +   shutdownProcessor := TRUE;
44  +end process;
```

CallHistory.tla v1->v2

```
15  (*--algorithm CallHistory
16
17  variables
18      queue = <<>>,
19      commited = 0,
20  +   shutdownProcessor = FALSE,
21      store = <<>>;
32
33  fair process Call \in Calls
34  begin CallSetup:
35      queue := Append(queue, [id |-> self, action |-> "setup"]);
36  CallComplete:
37      queue := Append(queue, [id |-> self, action |-> "complete"]);
38  end process;
39
40  +fair process Shutdown = "Shutdown"
41  +begin SignalShutdown:
42  +   when \A c \in Calls: pc[c] = "Done";
43  +   shutdownProcessor := TRUE;
44  +end process;
```

CallHistory.tla v1->v2

```
 begin Process:
     while TRUE do
+        either
             when Len(queue) >= offset;
 ProcessCDR:
             with cdr = queue[offset]
             do
                 if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
                     wip := Add(wip, cdr.id, [offset |-> offset]);
                 elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
                     wip := Remove(wip, cdr.id);
                     store := Append(store, cdr.id);
                 end if;
             end with;
             commited := offset;
             offset := offset + 1;
+        or
+            when shutdownProcessor;
+            goto Done;
+        end either;
     end while;
 end process;
```

CallHistory.tla v1->v2

```
 begin Process:
     while TRUE do
+        either
             when Len(queue) >= offset;
 ProcessCDR:
             with cdr = queue[offset]
             do
                 if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
                     wip := Add(wip, cdr.id, [offset |-> offset]);
                 elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
                     wip := Remove(wip, cdr.id);
                     store := Append(store, cdr.id);
                 end if;
             end with;
             commited := offset;
             offset := offset + 1;
+        or
+            when shutdownProcessor;
+            goto Done;
+        end either;
     end while;
 end process;
```

CallHistory.tla v1->v2

```
SPECIFICATION Spec
\* Add statements after this line.

CONSTANTS
    Calls = {"call-1", "call-2", "call-3"}

PROPERTIES
    Success
    Termination
```

CallHistory.tla v2 Run: Failed

```
21  Error: Temporal properties were violated.
22
23  Error: The following behavior constitutes a counter-example:
306
307 State 19: <Process line 120, col 12 to line 126, col 36 of module CallHistory>
308 /\ store = <<"call-3", "call-2">>
309 /\ wip = ("call-1" :> [offset |-> 5])
310 /\ pc = ( "call-1" :> "Done" @@
311   "call-2" :> "Done" @@
312   "call-3" :> "Done" @@
313   "Shutdown" :> "Done" @@
314   "Processor" :> "Done" )
315 /\ offset = 6
316 /\ shutdownProcessor = TRUE
317 /\ queue = << [id |-> "call-2", action |-> "setup"],
318    [id |-> "call-3", action |-> "setup"],
319    [id |-> "call-3", action |-> "complete"],
320    [id |-> "call-2", action |-> "complete"],
321    [id |-> "call-1", action |-> "setup"],
322    [id |-> "call-1", action |-> "complete"] >>
323 /\ commited = 5
```

CallHistory.tla v2 Run: Failed

```
21  Error: Temporal properties were violated.
22
23  Error: The following behavior constitutes a counter-example:
306
307 State 19: <Process line 120, col 12 to line 126, col 36 of module CallHistory>
308 /\ store = <<"call-3", "call-2">>
309 /\ wip = ("call-1" :> [offset |-> 5])
310 /\ pc = ( "call-1" :> "Done" @@
311   "call-2" :> "Done" @@
312   "call-3" :> "Done" @@
313   "Shutdown" :> "Done" @@
314   "Processor" :> "Done" )
315 /\ offset = 6
316 /\ shutdownProcessor = TRUE
317 /\ queue = << [id |-> "call-2", action |-> "setup"],
318    [id |-> "call-3", action |-> "setup"],
319    [id |-> "call-3", action |-> "complete"],
320    [id |-> "call-2", action |-> "complete"],
321    [id |-> "call-1", action |-> "setup"],
322    [id |-> "call-1", action |-> "complete"] >>
323 /\ commited = 5
```

CallHistory.tla v2 Run: Failed

```
21  Error: Temporal properties were violated.
22
23  Error: The following behavior constitutes a counter-example:
306
307 State 19: <Process line 120, col 12 to line 126, col 36 of module CallHistory>
308 /\ store = <<"call-3", "call-2">>
309 /\ wip = ("call-1" :> [offset |-> 5])
310 /\ pc = ( "call-1" :> "Done" @@
311   "call-2" :> "Done" @@
312   "call-3" :> "Done" @@
313   "Shutdown" :> "Done" @@
314   "Processor" :> "Done" )
315 /\ offset = 6
316 /\ shutdownProcessor = TRUE
317 /\ queue = << [id |-> "call-2", action |-> "setup"],
318    [id |-> "call-3", action |-> "setup"],
319    [id |-> "call-3", action |-> "complete"],
320    [id |-> "call-2", action |-> "complete"],
321    [id |-> "call-1", action |-> "setup"],
322    [id |-> "call-1", action |-> "complete"] >>
323 /\ commited = 5
```

CallHistory.tla v2 Run: Failed

```
21  Error: Temporal properties were violated.
22
23  Error: The following behavior constitutes a counter-example:
306
307 State 19: <Process line 120, col 12 to line 126, col 36 of module CallHistory>
308 /\ store = <<"call-3", "call-2">>
309 /\ wip = ("call-1" :> [offset |-> 5])
310 /\ pc = ( "call-1" :> "Done" @@
311   "call-2" :> "Done" @@
312   "call-3" :> "Done" @@
313   "Shutdown" :> "Done" @@
314   "Processor" :> "Done" )
315 /\ offset = 6
316 /\ shutdownProcessor = TRUE
317 /\ queue = << [id |-> "call-2", action |-> "setup"],
318    [id |-> "call-3", action |-> "setup"],
319    [id |-> "call-3", action |-> "complete"],
320    [id |-> "call-2", action |-> "complete"],
321    [id |-> "call-1", action |-> "setup"],
322    [id |-> "call-1", action |-> "complete"] >>
323 /\ commited = 5
```

CallHistory.tla v2->v3

```
50  begin Process:
51      while TRUE do
52          either
53              when Len(queue) >= offset;
54  ProcessCDR:
55              with cdr = queue[offset]
56              do
57                  if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
58                      wip := Add(wip, cdr.id, [offset |-> offset]);
59                  elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
60                      wip := Remove(wip, cdr.id);
61                      store := Append(store, cdr.id);
62                  end if;
63              end with;
64              commited := offset;
65              offset := offset + 1;
66          or
67  -           when shutdownProcessor;
68  +           when shutdownProcessor /\ wip = <<>>;
69              goto Done;
70          end either;
71      end while;
```

CallHistory.tla v3 Run: Failed

```
21  Error: Temporal properties were violated.
22
23  Error: The following behavior constitutes a counter-example:
24
148
149 State 9: <Process line 120, col 12 to line 126, col 36 of module CallHistory>
150 /\ store = <<>>
151 /\ wip = <<>>
152 /\ pc = ( "call-1" :> "Done" @@
153   "call-2" :> "Done" @@
154   "call-3" :> "Done" @@
155   "Shutdown" :> "Done" @@
156   "Processor" :> "Done" )
157 /\ offset = 1
158 /\ shutdownProcessor = TRUE
159 /\ queue = << [id |-> "call-2", action |-> "setup"],
160    [id |-> "call-2", action |-> "complete"],
161    [id |-> "call-1", action |-> "setup"],
162    [id |-> "call-1", action |-> "complete"],
163    [id |-> "call-3", action |-> "setup"],
164    [id |-> "call-3", action |-> "complete"] >>
165 /\ commited = 0
```

CallHistory.tla v3 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:


State 9: <Process line 120, col 12 to line 126, col 36 of module CallHistory>
/\ store = <<>>
/\ wip = <<>>
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 1
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-2", action |-> "setup"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 0
```

CallHistory.tla v3 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:


State 9: <Process line 120, col 12 to line 126, col 36 of module CallHistory>
/\ store = <<>>
/\ wip = <<>>
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 1
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-2", action |-> "setup"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 0
```

CallHistory.tla v3 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:


State 9: <Process line 120, col 12 to line 126, col 36 of module CallHistory>
/\ store = <<>>
/\ wip = <<>>
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 1
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-2", action |-> "setup"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 0
```

CallHistory.tla v3->v4

```
54  ProcessCDR:
55              with cdr = queue[offset]
56              do
57                  if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
58                      wip := Add(wip, cdr.id, [offset |-> offset]);
59                  elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
60                      wip := Remove(wip, cdr.id);
61                      store := Append(store, cdr.id);
62                  end if;
63              end with;
64              commited := offset;
65              offset := offset + 1;
66          or
67  -           when shutdownProcessor /\ wip = <<>>;
68  +           when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
69              goto Done;
70          end either;
71      end while;
72  end process;
```

CallHistory.tla v4 Run: Succeed

```
22 Model checking completed. No error has been found.
23   Estimates of the probability that TLC did not check all reachable states
24   because two distinct states had the same fingerprint:
25   calculated (optimistic):  val = 7.2E-13
26 7357 states generated, 4129 distinct states found, 0 states left on queue.
27 The depth of the complete state graph search is 21.
```

CallHistory.tla v4 Run: Succeed

```
22 Model checking completed. No error has been found.
23   Estimates of the probability that TLC did not check all reachable states
24   because two distinct states had the same fingerprint:
25   calculated (optimistic):  val = 7.2E-13
26 7357 states generated, 4129 distinct states found, 0 states left on queue.
27 The depth of the complete state graph search is 21.
```

Модель:
добавим аварийный останов

CallHistory.tla v4->v5

```
47    fair process Processor = "Processor"
48    variables
49        wip = <<>>,
50  +     crashes = 0,
51        offset = commited + 1;
52    begin Process:
53        while TRUE do
54            either
55                when Len(queue) >= offset;
56    ProcessCDR:
```

```
71  +         or
72  +             when crashes < Crashes;
73  +Crash:
74  +             crashes := crashes + 1 ||
75  +             offset := commited + 1 ||
76  +             wip := <<>>;
```

CallHistory.tla v4->v5

```
47    fair process Processor = "Processor"
48    variables
49        wip = <<>>,
50  +     crashes = 0,
51        offset = commited + 1;
52    begin Process:
53        while TRUE do
54            either
55                when Len(queue) >= offset;
56    ProcessCDR:
```

```
71  +         or
72  +             when crashes < Crashes;
73  +Crash:
74  +             crashes := crashes + 1 ||
75  +             offset := commited + 1 ||
76  +             wip := <<>>;
```

CallHistory.tla v4->v5

```
SPECIFICATION Spec
\* Add statements after this line.

CONSTANTS
    Calls = {"call-1", "call-2", "call-3"}
    Crashes = 1

PROPERTIES
    Success
    Termination
```

CallHistory.tla v5 Run: Failed

```
21  Error: Temporal properties were violated.
22
23  Error: The following behavior constitutes a counter-example:
405
406 State 23: <Process line 130, col 12 to line 138, col 45 of module CallHistory>
407 /\ store = <<"call-2", "call-3">>
408 /\ wip = << >>
409 /\ crashes = 1
410 /\ pc = ( "call-1" :> "Done" @@
411   "call-2" :> "Done" @@
412   "call-3" :> "Done" @@
413   "Shutdown" :> "Done" @@
414   "Processor" :> "Done" )
415 /\ offset = 7
416 /\ shutdownProcessor = TRUE
417 /\ queue = << [id |-> "call-1", action |-> "setup"],
418    [id |-> "call-2", action |-> "setup"],
419    [id |-> "call-3", action |-> "setup"],
420    [id |-> "call-1", action |-> "complete"],
421    [id |-> "call-2", action |-> "complete"],
422    [id |-> "call-3", action |-> "complete"] >>
423 /\ commited = 6
424
425 State 24: Stuttering
```

CallHistory.tla v5 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 23: <Process line 130, col 12 to line 138, col 45 of module CallHistory>
/\ store = <<"call-2", "call-3">>
/\ wip = << >>
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 7
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-1", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 6

State 24: Stuttering
```

CallHistory.tla v5 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 23: <Process line 130, col 12 to line 138, col 45 of module CallHistory>
/\ store = <<"call-2", "call-3">>
/\ wip = << >>
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 7
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-1", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 6

State 24: Stuttering
```

CallHistory.tla v5 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 23: <Process line 130, col 12 to line 138, col 45 of module CallHistory>
/\ store = <<"call-2", "call-3">>
/\ wip = << >>
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 7
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-1", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 6

State 24: Stuttering
```

CallHistory.tla v5 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 23: <Process line 130, col 12 to line 138, col 45 of module CallHistory>
/\ store = <<"call-2", "call-3">>
/\ wip = << >>
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 7
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-1", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 6

State 24: Stuttering
```

CallHistory.tla v5 Run: Failed

```
State 9: <CallComplete line 114, col 23 to line 118, col 56 of module CallHistory>
/\ store = <<>>
/\ wip = ("call-1" :> [offset |-> 1])
/\ crashes = 0
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "CallComplete" @@
  "Shutdown" :> "SignalShutdown" @@
  "Processor" :> "Crash" )
/\ offset = 2
/\ shutdownProcessor = FALSE
/\ queue = << [id |-> "call-1", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"] >>
/\ commited = 1
```

CallHistory.tla v5 Run: Failed

```
State 9: <CallComplete line 114, col 23 to line 118, col 56 of module CallHistory>
/\ store = <<>>
/\ wip = ("call-1" :> [offset |-> 1])
/\ crashes = 0
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "CallComplete" @@
  "Shutdown" :> "SignalShutdown" @@
  "Processor" :> "Crash" )
/\ offset = 2
/\ shutdownProcessor = FALSE
/\ queue = << [id |-> "call-1", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"] >>
/\ commited = 1
```

CallHistory.tla v5 Run: Failed

```
State 9: <CallComplete line 114, col 23 to line 118, col 56 of module CallHistory>
/\ store = <<>>
/\ wip = ("call-1" :> [offset |-> 1])
/\ crashes = 0
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "CallComplete" @@
  "Shutdown" :> "SignalShutdown" @@
  "Processor" :> "Crash" )
/\ offset = 2
/\ shutdownProcessor = FALSE
/\ queue = << [id |-> "call-1", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"] >>
/\ commited = 1
```

CallHistory.tla v5 Run: Failed

```
State 10: <Crash line 155, col 10 to line 160, col 69 of module CallHistory>
/\ store = <<>>
/\ wip = <<>>
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "CallComplete" @@
  "Shutdown" :> "SignalShutdown" @@
  "Processor" :> "Process" )
/\ offset = 2
/\ shutdownProcessor = FALSE
/\ queue = << [id |-> "call-1", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"] >>
/\ commited = 1
```

CallHistory.tla v5->v6

```
13 | +set ++ x == set \union {x}

58   fair process Processor = "Processor"
59   variables
60       wip = <<>>,
61 +     processed = {},
62       crashes = 0,
63       offset = commited + 1;
64   begin Process:
65       while TRUE do
66           either
67               when Len(queue) >= offset;
68   ProcessCDR:
69               with cdr = queue[offset]
70               do
71                   if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
72                       wip := Add(wip, cdr.id, [offset |-> offset]);
73                   elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
74 +                     processed := processed ++ wip[cdr.id].offset ++ offset;
75                       wip := Remove(wip, cdr.id);
76 +                     commited := CommitOffset(processed, commited);
77                       store := Append(store, cdr.id);
78                   end if;
79               end with;
80 -             commited := offset;
```

CallHistory.tla v5->v6

```
13 | +set ++ x == set \union {x}

58   fair process Processor = "Processor"
59   variables
60       wip = <<>>,
61 +     processed = {},
62       crashes = 0,
63       offset = commited + 1;
64   begin Process:
65       while TRUE do
66           either
67               when Len(queue) >= offset;
68   ProcessCDR:
69               with cdr = queue[offset]
70               do
71                   if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
72                       wip := Add(wip, cdr.id, [offset |-> offset]);
73                   elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
74 +                     processed := processed ++ wip[cdr.id].offset ++ offset;
75                       wip := Remove(wip, cdr.id);
76 +                     commited := CommitOffset(processed, commited);
77                       store := Append(store, cdr.id);
78                   end if;
79               end with;
80 -             commited := offset;
```

CallHistory.tla v5->v6

```
13 +set ++ x == set \union {x}

58  fair process Processor = "Processor"
59  variables
60      wip = <<>>,
61 +    processed = {},
62      crashes = 0,
63      offset = commited + 1;
64  begin Process:
65      while TRUE do
66          either
67              when Len(queue) >= offset;
68  ProcessCDR:
69              with cdr = queue[offset]
70              do
71                  if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
72                      wip := Add(wip, cdr.id, [offset |-> offset]);
73                  elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
74 +                    processed := processed ++ wip[cdr.id].offset ++ offset;
75                      wip := Remove(wip, cdr.id);
76 +                    commited := CommitOffset(processed, commited);
77                      store := Append(store, cdr.id);
78                  end if;
79              end with;
80 -            commited := offset;
```

CallHistory.tla v5->v6

```
81              offset := offset + 1;
82          or
83              when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
84              goto Done;
85          or
86              when crashes < Crashes;
87  Crash:
88              crashes := crashes + 1 ||
89  +           processed := {} ||
90              offset := commited + 1 ||
91              wip := <<>>;
92          end either;
93      end while;
94  end process;
```

CallHistory.tla v5->v6

```
24   define
25
26  +    \* {1, 3}, 0 -> 0
27  +    \* {2, 3}, 0 -> 0
28  +    \* {1, 2, 3}, 0 -> 3
```

CallHistory.tla v5->v6

```
 define

+    \* {1, 3}, 0 -> 0
+    \* {2, 3}, 0 -> 0
+    \* {1, 2, 3}, 0 -> 3
+    CommitOffset(set, current) ==
+        LET first == current + 1 IN
+        IF /\ first \in set
+           /\ \A i \in {j \in set: j > first}: i-1 \in set
+        THEN CHOOSE i \in set: i+1 \notin set
+        ELSE current
+
     AllCallsProcessed == <>[](
         \A i \in Calls:
            Len(SelectSeq(store, LAMBDA j: j = i)) = 1
     )

     Success == AllCallsProcessed

 end define;
```

CallHistory.tla v5->v6

```
24  define
25
26  +    \* {1, 3}, 0 -> 0
27  +    \* {2, 3}, 0 -> 0
28  +    \* {1, 2, 3}, 0 -> 3
29  +    CommitOffset(set, current) ==
30  +        LET first == current + 1 IN
31  +        IF /\ first \in set
32  +           /\ \A i \in {j \in set: j > first}: i-1 \in set
33  +        THEN CHOOSE i \in set: i+1 \notin set
34  +        ELSE current
35  +
36      AllCallsProcessed == <>[](
37          \A i \in Calls:
38             Len(SelectSeq(store, LAMBDA j: j = i)) = 1
39      )
40
41      Success == AllCallsProcessed
42
43  end define;
```

CallHistory.tla v5->v6

```
 define

+    \* {1, 3}, 0 -> 0
+    \* {2, 3}, 0 -> 0
+    \* {1, 2, 3}, 0 -> 3
+    CommitOffset(set, current) ==
+        LET first == current + 1 IN
+        IF /\ first \in set
+           /\ \A i \in {j \in set: j > first}: i-1 \in set
+        THEN CHOOSE i \in set: i+1 \notin set
+        ELSE current
+
     AllCallsProcessed == <>[](
         \A i \in Calls:
            Len(SelectSeq(store, LAMBDA j: j = i)) = 1
     )

     Success == AllCallsProcessed

 end define;
```

CallHistory.tla v5->v6

```
24   define
25
26  +    \* {1, 3}, 0 -> 0
27  +    \* {2, 3}, 0 -> 0
28  +    \* {1, 2, 3}, 0 -> 3
29  +    CommitOffset(set, current) ==
30  +        LET first == current + 1 IN
31  +        IF /\ first \in set
32  +           /\ \A i \in {j \in set: j > first}: i-1 \in set
33  +        THEN CHOOSE i \in set: i+1 \notin set
34  +        ELSE current
35  +
36       AllCallsProcessed == <>[](
37           \A i \in Calls:
38              Len(SelectSeq(store, LAMBDA j: j = i)) = 1
39       )
40
41       Success == AllCallsProcessed
42
43   end define;
```

CallHistory.tla v5->v6

```
 define

+    \* {1, 3}, 0 -> 0
+    \* {2, 3}, 0 -> 0
+    \* {1, 2, 3}, 0 -> 3
+    CommitOffset(set, current) ==
+        LET first == current + 1 IN
+        IF /\ first \in set
+           /\ \A i \in {j \in set: j > first}: i-1 \in set
+        THEN CHOOSE i \in set: i+1 \notin set
+        ELSE current
+
     AllCallsProcessed == <>[](
         \A i \in Calls:
            Len(SelectSeq(store, LAMBDA j: j = i)) = 1
     )

     Success == AllCallsProcessed

 end define;
```

CallHistory.tla v6 Run: Failed

```
Error: The following behavior constitutes a counter-example:

State 29: <Process line 153, col 12 to line 161, col 56 of module CallHistory>
/\ store = <<"call-1", "call-1", "call-2", "call-3">>
/\ processed = {1, 2, 3, 4, 5, 6}
/\ wip = << >>
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 7
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-2", action |-> "setup"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 6

State 30: Stuttering
```

CallHistory.tla v6 Run: Failed

```
Error: The following behavior constitutes a counter-example:

State 29: <Process line 153, col 12 to line 161, col 56 of module CallHistory>
/\ store = <<"call-1", "call-1", "call-2", "call-3">>
/\ processed = {1, 2, 3, 4, 5, 6}
/\ wip = << >>
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 7
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-2", action |-> "setup"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 6

State 30: Stuttering
```

CallHistory.tla v6 Run: Failed

```
Error: The following behavior constitutes a counter-example:

State 29: <Process line 153, col 12 to line 161, col 56 of module CallHistory>
/\ store = <<"call-1", "call-1", "call-2", "call-3">>
/\ processed = {1, 2, 3, 4, 5, 6}
/\ wip = << >>
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Done" )
/\ offset = 7
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-2", action |-> "setup"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-3", action |-> "complete"] >>
/\ commited = 6

State 30: Stuttering
```

CallHistory.tla v6->v7

```
19  variables
20      queue = <<>>,
21      commited = 0,
22      shutdownProcessor = FALSE,
23  -   store = <<>>;
24  +   store = {};

36
37  -   AllCallsProcessed == <>[](
38  -       \A i \in Calls:
39  -          Len(SelectSeq(store, LAMBDA j: j = i)) = 1
40  -   )
41  +   AllCallsProcessed == <>[](Calls = store)
42

75                  elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
76                      processed := processed ++ wip[cdr.id].offset ++ offset;
77                      wip := Remove(wip, cdr.id);
78                      commited := CommitOffset(processed, commited);
79  -                   store := Append(store, cdr.id);
80  +                   store := store ++ cdr.id;
81                  end if;
```

CallHistory.tla v6->v7

```
19  variables
20      queue = <<>>,
21      commited = 0,
22      shutdownProcessor = FALSE,
23  -   store = <<>>;
24  +   store = {};

36
37  -   AllCallsProcessed == <>[](
38  -       \A i \in Calls:
39  -          Len(SelectSeq(store, LAMBDA j: j = i)) = 1
40  -   )
41  +   AllCallsProcessed == <>[](Calls = store)
42

75                  elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
76                      processed := processed ++ wip[cdr.id].offset ++ offset;
77                      wip := Remove(wip, cdr.id);
78                      commited := CommitOffset(processed, commited);
79  -                   store := Append(store, cdr.id);
80  +                   store := store ++ cdr.id;
81                  end if;
```

CallHistory.tla v6->v7

```
19   variables
20       queue = <<>>,
21       commited = 0,
22       shutdownProcessor = FALSE,
23  -    store = <<>>;
24  +    store = {};
36
37  -    AllCallsProcessed == <>[](
38  -        \A i \in Calls:
39  -            Len(SelectSeq(store, LAMBDA j: j = i)) = 1
40  -    )
41  +    AllCallsProcessed == <>[](Calls = store)
42
75                       elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
76                           processed := processed ++ wip[cdr.id].offset ++ offset;
77                           wip := Remove(wip, cdr.id);
78                           commited := CommitOffset(processed, commited);
79  -                        store := Append(store, cdr.id);
80  +                        store := store ++ cdr.id;
81                       end if;
```

CallHistory.tla v7 Run: Succeed

```
22 Model checking completed. No error has been found.
23   Estimates of the probability that TLC did not check all reachable states
24   because two distinct states had the same fingerprint:
25   calculated (optimistic):  val = 1.2E-11
26 29490 states generated, 15435 distinct states found, 0 states left on queue.
27 The depth of the complete state graph search is 29.
```

CallHistory.tla v7 Run: Succeed

```
22 Model checking completed. No error has been found.
23   Estimates of the probability that TLC did not check all reachable states
24   because two distinct states had the same fingerprint:
25   calculated (optimistic):  val = 1.2E-11
26 29490 states generated, 15435 distinct states found, 0 states left on queue.
27 The depth of the complete state graph search is 29.
```

Модель:
добавим аварийный сброс вызова

CallHistory.tla v7->v8

```
42   fair process Call \in Calls
43   begin CallSetup:
44       queue := Append(queue, [id |-> self, action |-> "setup"]);
45   CallComplete:
46  +    either
47           queue := Append(queue, [id |-> self, action |-> "complete"]);
48  +    or
49  +        skip;
50  +    end either;
51   end process;
```

CallHistory.tla v7->v8

```
42    fair process Call \in Calls
43    begin CallSetup:
44        queue := Append(queue, [id |-> self, action |-> "setup"]);
45    CallComplete:
46  +     either
47            queue := Append(queue, [id |-> self, action |-> "complete"]);
48  +     or
49  +         skip;
50  +     end either;
51    end process;
```

CallHistory.tla v8 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 20: <ProcessCDR line 163, col 15 to line 178, col 66 of module CallHistory>
/\ store = {"call-1", "call-3"}
/\ processed = {1, 2, 3, 4}
/\ wip = ("call-2" :> [offset |-> 5])
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Process" )
/\ offset = 6
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-3", action |-> "setup"],
   [id |-> "call-3", action |-> "complete"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "setup"] >>
/\ commited = 4

State 21: Stuttering
```

CallHistory.tla v8 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 20: <ProcessCDR line 163, col 15 to line 178, col 66 of module CallHistory>
/\ store = {"call-1", "call-3"}
/\ processed = {1, 2, 3, 4}
/\ wip = ("call-2" :> [offset |-> 5])
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Process" )
/\ offset = 6
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-3", action |-> "setup"],
   [id |-> "call-3", action |-> "complete"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "setup"] >>
/\ commited = 4

State 21: Stuttering
```

CallHistory.tla v8 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 20: <ProcessCDR line 163, col 15 to line 178, col 66 of module CallHistory>
/\ store = {"call-1", "call-3"}
/\ processed = {1, 2, 3, 4}
/\ wip = ("call-2" :> [offset |-> 5])
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Process" )
/\ offset = 6
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-3", action |-> "setup"],
   [id |-> "call-3", action |-> "complete"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"],
   [id |-> "call-2", action |-> "setup"] >>
/\ commited = 4

State 21: Stuttering
```

CallHistory.tla v8->v9

```
19    variables
20        queue = <<>>,
21        commited = 0,
22        shutdownProcessor = FALSE,
23  +     failed = {},
24        store = {};

37  -     AllCallsProcessed == <>[](Calls = store)
38  +     AllCallsProcessed == <>[](Calls \ failed = store)
39
40        Success == AllCallsProcessed

44    fair process Call \in Calls
45    begin CallSetup:
46        queue := Append(queue, [id |-> self, action |-> "setup"]);
47    CallComplete:
48        either
49            queue := Append(queue, [id |-> self, action |-> "complete"]);
50        or
51  -         skip;
52  +         failed := failed ++ self;
53        end either;
54    end process;
```

CallHistory.tla v8->v9

```
19    variables
20        queue = <<>>,
21        commited = 0,
22        shutdownProcessor = FALSE,
23  +     failed = {},
24        store = {};

37  -     AllCallsProcessed == <>[](Calls = store)
38  +     AllCallsProcessed == <>[](Calls \ failed = store)
39
40        Success == AllCallsProcessed

44    fair process Call \in Calls
45    begin CallSetup:
46        queue := Append(queue, [id |-> self, action |-> "setup"]);
47    CallComplete:
48        either
49            queue := Append(queue, [id |-> self, action |-> "complete"]);
50        or
51  -         skip;
52  +         failed := failed ++ self;
53        end either;
54    end process;
```

CallHistory.tla v8->v9

```
19    variables
20        queue = <<>>,
21        commited = 0,
22        shutdownProcessor = FALSE,
23  +     failed = {},
24        store = {};

37  -     AllCallsProcessed == <>[](Calls = store)
38  +     AllCallsProcessed == <>[](Calls \ failed = store)
39
40        Success == AllCallsProcessed

44    fair process Call \in Calls
45    begin CallSetup:
46        queue := Append(queue, [id |-> self, action |-> "setup"]);
47    CallComplete:
48        either
49            queue := Append(queue, [id |-> self, action |-> "complete"]);
50        or
51  -         skip;
52  +         failed := failed ++ self;
53        end either;
54    end process;
```

CallHistory.tla v9 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 18: <ProcessCDR line 166, col 15 to line 181, col 74 of module CallHistory>
/\ store = {"call-1"}
/\ processed = {3, 4}
/\ wip = ("call-2" :> [offset |-> 1] @@ "call-3" :> [offset |-> 2])
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Process" )
/\ offset = 5
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"] >>
/\ commited = 0
/\ failed = {"call-2", "call-3"}

State 19: Stuttering
```

CallHistory.tla v9 Run: Failed

```
21  Error: Temporal properties were violated.
22
23  Error: The following behavior constitutes a counter-example:
317
318 State 18: <ProcessCDR line 166, col 15 to line 181, col 74 of module CallHistory>
319 /\ store = {"call-1"}
320 /\ processed = {3, 4}
321 /\ wip = ("call-2" :> [offset |-> 1] @@ "call-3" :> [offset |-> 2])
322 /\ crashes = 1
323 /\ pc = ( "call-1" :> "Done" @@
324   "call-2" :> "Done" @@
325   "call-3" :> "Done" @@
326   "Shutdown" :> "Done" @@
327   "Processor" :> "Process" )
328 /\ offset = 5
329 /\ shutdownProcessor = TRUE
330 /\ queue = << [id |-> "call-2", action |-> "setup"],
331    [id |-> "call-3", action |-> "setup"],
332    [id |-> "call-1", action |-> "setup"],
333    [id |-> "call-1", action |-> "complete"] >>
334 /\ commited = 0
335 /\ failed = {"call-2", "call-3"}
336
337 State 19: Stuttering
```

CallHistory.tla v9 Run: Failed

```
Error: Temporal properties were violated.

Error: The following behavior constitutes a counter-example:

State 18: <ProcessCDR line 166, col 15 to line 181, col 74 of module CallHistory>
/\ store = {"call-1"}
/\ processed = {3, 4}
/\ wip = ("call-2" :> [offset |-> 1] @@ "call-3" :> [offset |-> 2])
/\ crashes = 1
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Process" )
/\ offset = 5
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-2", action |-> "setup"],
   [id |-> "call-3", action |-> "setup"],
   [id |-> "call-1", action |-> "setup"],
   [id |-> "call-1", action |-> "complete"] >>
/\ commited = 0
/\ failed = {"call-2", "call-3"}

State 19: Stuttering
```

CallHistory.tla v9->v10

```
10  CONSTANTS Calls
11  CONSTANTS Crashes
12 +CONSTANTS Timeout

62  fair process Processor = "Processor"
63  variables
64      wip = <<>>,
65      processed = {},
66      crashes = 0,
67 +    now = 0,
68      offset = commited + 1;
69  begin Process:
70      while TRUE do
71 +        now := now + 1;
72          either
73              when Len(queue) >= offset;
74  ProcessCDR:
75              with cdr = queue[offset]
76              do
77                  if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
78 -                    wip := Add(wip, cdr.id, [offset |-> offset]);
79 +                    wip := Add(wip, cdr.id, [start |-> now, offset |-> offset]);
80                  elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
```

CallHistory.tla v9->v10

```
 CONSTANTS Calls
 CONSTANTS Crashes
+CONSTANTS Timeout
```

```
 fair process Processor = "Processor"
 variables
     wip = <<>>,
     processed = {},
     crashes = 0,
+    now = 0,
     offset = commited + 1;
 begin Process:
     while TRUE do
+        now := now + 1;
         either
             when Len(queue) >= offset;
 ProcessCDR:
             with cdr = queue[offset]
             do
                 if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
-                    wip := Add(wip, cdr.id, [offset |-> offset]);
+                    wip := Add(wip, cdr.id, [start |-> now, offset |-> offset]);
                 elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
```

CallHistory.tla v9->v10

```
10   CONSTANTS Calls
11   CONSTANTS Crashes
12  +CONSTANTS Timeout
```

```
62   fair process Processor = "Processor"
63   variables
64       wip = <<>>,
65       processed = {},
66       crashes = 0,
67  +    now = 0,
68       offset = commited + 1;
69   begin Process:
70       while TRUE do
71  +        now := now + 1;
72           either
73               when Len(queue) >= offset;
74   ProcessCDR:
75               with cdr = queue[offset]
76               do
77                   if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
78  -                    wip := Add(wip, cdr.id, [offset |-> offset]);
79  +                    wip := Add(wip, cdr.id, [start |-> now, offset |-> offset]);
80                   elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
```

CallHistory.tla v9->v10

```
 CONSTANTS Calls
 CONSTANTS Crashes
+CONSTANTS Timeout
```

```
 fair process Processor = "Processor"
 variables
     wip = <<>>,
     processed = {},
     crashes = 0,
+    now = 0,
     offset = commited + 1;
 begin Process:
     while TRUE do
+        now := now + 1;
         either
             when Len(queue) >= offset;
 ProcessCDR:
             with cdr = queue[offset]
             do
                 if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
-                    wip := Add(wip, cdr.id, [offset |-> offset]);
+                    wip := Add(wip, cdr.id, [start |-> now, offset |-> offset]);
                 elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
```

CallHistory.tla v9->v10

```
 CallComplete:
     either
         queue := Append(queue, [id |-> self, action |-> "complete"]);
     or
-        failed := failed ++ self;
+        skip;
     end either;
```

```
         or
             when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
             goto Done;
+        or
+            when \E id \in DOMAIN wip: (now - wip[id].start) >= Timeout;
+Cleanup:
+            with
+                id = CHOOSE id \in DOMAIN wip: (now - wip[id].start) >= Timeout,
+                off = wip[id].offset
+            do
+                processed := processed ++ off;
+                wip := Remove(wip, id);
+                commited := CommitOffset(processed, commited);
+                failed := failed ++ id;
+            end with;
         or
```

CallHistory.tla v9->v10

```
47   CallComplete:
48       either
49           queue := Append(queue, [id |-> self, action |-> "complete"]);
50       or
51   -       failed := failed ++ self;
52   +       skip;
53       end either;
```

```
88           or
89               when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
90               goto Done;
91   +       or
92   +           when \E id \in DOMAIN wip: (now - wip[id].start) >= Timeout;
93   +Cleanup:
94   +           with
95   +               id = CHOOSE id \in DOMAIN wip: (now - wip[id].start) >= Timeout,
96   +               off = wip[id].offset
97   +           do
98   +               processed := processed ++ off;
99   +               wip := Remove(wip, id);
100  +               commited := CommitOffset(processed, commited);
101  +               failed := failed ++ id;
102  +           end with;
103          or
```

CallHistory.tla v9->v10

```
47  CallComplete:
48      either
49          queue := Append(queue, [id |-> self, action |-> "complete"]);
50      or
51  -       failed := failed ++ self;
52  +       skip;
53      end either;
```

```
88          or
89              when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
90              goto Done;
91  +       or
92  +           when \E id \in DOMAIN wip: (now - wip[id].start) >= Timeout;
93  +Cleanup:
94  +           with
95  +               id = CHOOSE id \in DOMAIN wip: (now - wip[id].start) >= Timeout,
96  +               off = wip[id].offset
97  +           do
98  +               processed := processed ++ off;
99  +               wip := Remove(wip, id);
100 +               commited := CommitOffset(processed, commited);
101 +               failed := failed ++ id;
102 +           end with;
103         or
```

CallHistory.tla v9->v10

```
47   CallComplete:
48       either
49           queue := Append(queue, [id |-> self, action |-> "complete"]);
50       or
51   -       failed := failed ++ self;
52   +       skip;
53       end either;
```

```
88           or
89               when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
90               goto Done;
91   +       or
92   +           when \E id \in DOMAIN wip: (now - wip[id].start) >= Timeout;
93   +Cleanup:
94   +           with
95   +               id = CHOOSE id \in DOMAIN wip: (now - wip[id].start) >= Timeout,
96   +               off = wip[id].offset
97   +           do
98   +               processed := processed ++ off;
99   +               wip := Remove(wip, id);
100  +               commited := CommitOffset(processed, commited);
101  +               failed := failed ++ id;
102  +           end with;
103          or
```

CallHistory.tla v9->v10

```
47  CallComplete:
48      either
49          queue := Append(queue, [id |-> self, action |-> "complete"]);
50      or
51  -           failed := failed ++ self;
52  +           skip;
53      end either;
```

```
88          or
89              when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
90              goto Done;
91  +       or
92  +           when \E id \in DOMAIN wip: (now - wip[id].start) >= Timeout;
93  +Cleanup:
94  +           with
95  +               id = CHOOSE id \in DOMAIN wip: (now - wip[id].start) >= Timeout,
96  +               off = wip[id].offset
97  +           do
98  +               processed := processed ++ off;
99  +               wip := Remove(wip, id);
100 +               commited := CommitOffset(processed, commited);
101 +               failed := failed ++ id;
102 +           end with;
103         or
```

CallHistory.tla v9->v10

```
47  CallComplete:
48      either
49          queue := Append(queue, [id |-> self, action |-> "complete"]);
50      or
51  -           failed := failed ++ self;
52  +           skip;
53      end either;
```

```
88          or
89              when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
90              goto Done;
91  +       or
92  +           when \E id \in DOMAIN wip: (now - wip[id].start) >= Timeout;
93  +Cleanup:
94  +           with
95  +               id = CHOOSE id \in DOMAIN wip: (now - wip[id].start) >= Timeout,
96  +               off = wip[id].offset
97  +           do
98  +               processed := processed ++ off;
99  +               wip := Remove(wip, id);
100 +               commited := CommitOffset(processed, commited);
101 +               failed := failed ++ id;
102 +           end with;
103         or
```

CallHistory.tla v9->v10

```
47  CallComplete:
48      either
49          queue := Append(queue, [id |-> self, action |-> "complete"]);
50      or
51  -           failed := failed ++ self;
52  +           skip;
53      end either;
```

```
88          or
89              when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
90              goto Done;
91  +       or
92  +           when \E id \in DOMAIN wip: (now - wip[id].start) >= Timeout;
93  +Cleanup:
94  +           with
95  +               id = CHOOSE id \in DOMAIN wip: (now - wip[id].start) >= Timeout,
96  +               off = wip[id].offset
97  +           do
98  +               processed := processed ++ off;
99  +               wip := Remove(wip, id);
100 +               commited := CommitOffset(processed, commited);
101 +               failed := failed ++ id;
102 +           end with;
103         or
```

CallHistory.tla v9->v10

```
SPECIFICATION Spec
\* Add statements after this line.

CONSTANTS
    Calls = {"call-1", "call-2", "call-3"}
    Crashes = 1
    Timeout = 10

PROPERTIES
    Success
    Termination
```

CallHistory.tla v10 Run: Failed

```
Error: The following behavior constitutes a counter-example:

State 26: <ProcessCDR line 185, col 15 to line 200, col 79 of module CallHistory>
/\ offset = 6
/\ commited = 4
/\ store = {"call-2", "call-3"}
/\ now = 9
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Process" )
/\ wip = ("call-1" :> [offset |-> 5, start |-> 9])
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-3", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "complete"],
   [id |-> "call-1", action |-> "setup"] >>
/\ crashes = 1
/\ failed = {}
/\ processed = {1, 2, 3, 4}

State 27: Stuttering
```

CallHistory.tla v10 Run: Failed

```
Error: The following behavior constitutes a counter-example:

State 26: <ProcessCDR line 185, col 15 to line 200, col 79 of module CallHistory>
/\ offset = 6
/\ commited = 4
/\ store = {"call-2", "call-3"}
/\ now = 9
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Process" )
/\ wip = ("call-1" :> [offset |-> 5, start |-> 9])
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-3", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "complete"],
   [id |-> "call-1", action |-> "setup"] >>
/\ crashes = 1
/\ failed = {}
/\ processed = {1, 2, 3, 4}

State 27: Stuttering
```

CallHistory.tla v10 Run: Failed

```
Error: The following behavior constitutes a counter-example:

State 26: <ProcessCDR line 185, col 15 to line 200, col 79 of module CallHistory>
/\ offset = 6
/\ commited = 4
/\ store = {"call-2", "call-3"}
/\ now = 9
/\ pc = ( "call-1" :> "Done" @@
  "call-2" :> "Done" @@
  "call-3" :> "Done" @@
  "Shutdown" :> "Done" @@
  "Processor" :> "Process" )
/\ wip = ("call-1" :> [offset |-> 5, start |-> 9])
/\ shutdownProcessor = TRUE
/\ queue = << [id |-> "call-3", action |-> "setup"],
   [id |-> "call-2", action |-> "setup"],
   [id |-> "call-2", action |-> "complete"],
   [id |-> "call-3", action |-> "complete"],
   [id |-> "call-1", action |-> "setup"] >>
/\ crashes = 1
/\ failed = {}
/\ processed = {1, 2, 3, 4}

State 27: Stuttering
```

CallHistory.tla v10->v11

```
86          or
87              when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
88              goto Done;
89  +       or
90  +           when /\ shutdownProcessor
91  +                /\ offset > Len(queue)
92  +                /\ wip /= <<>>
93  +                /\ \A id \in DOMAIN wip: (now - wip[id].start) < Timeout;
94  +           skip;
95          or
```

CallHistory.tla v10->v11

```
86          or
87              when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
88              goto Done;
89  +       or
90  +           when /\ shutdownProcessor
91  +                /\ offset > Len(queue)
92  +                /\ wip /= <<>>
93  +                /\ \A id \in DOMAIN wip: (now - wip[id].start) < Timeout;
94  +           skip;
95          or
```

CallHistory.tla v10->v11

```
86          or
87              when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
88              goto Done;
89  +       or
90  +           when /\ shutdownProcessor
91  +                /\ offset > Len(queue)
92  +                /\ wip /= <<>>
93  +                /\ \A id \in DOMAIN wip: (now - wip[id].start) < Timeout;
94  +           skip;
95          or
```

CallHistory.tla v11 Run: Succeed

```
25 Model checking completed. No error has been found.
26   Estimates of the probability that TLC did not check all reachable states
27   because two distinct states had the same fingerprint:
28   calculated (optimistic):  val = 2.7E-10
29   based on the actual fingerprints:  val = 1.5E-8
30 144036 states generated, 83106 distinct states found, 0 states left on queue.
31 The depth of the complete state graph search is 49.
```

CallHistory.tla v11 Run: Succeed

```
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
  calculated (optimistic):  val = 2.7E-10
  based on the actual fingerprints:  val = 1.5E-8
144036 states generated, 83106 distinct states found, 0 states left on queue.
The depth of the complete state graph search is 49.
```

CallHistory.tla v11 Run: Succeed

```
25 Model checking completed. No error has been found.
26   Estimates of the probability that TLC did not check all reachable states
27   because two distinct states had the same fingerprint:
28   calculated (optimistic):  val = 2.7E-10
29   based on the actual fingerprints:  val = 1.5E-8
30 144036 states generated, 83106 distinct states found, 0 states left on queue.
31 The depth of the complete state graph search is 49.
```

Модель:
итог

CallHistory.tla v11

```
------ MODULE CallHistory ------

EXTENDS Sequences, Integers, TLC

CONSTANTS Calls
CONSTANTS Crashes
CONSTANTS Timeout

set ++ x == set \union {x}
Add(dict, key, val) == dict @@ (key :> val)
Remove(dict, key) == [i \in DOMAIN dict \ {key} |-> dict[i]]

(*--algorithm CallHistory

variables
    queue = <<>>,
    commited = 0,
    shutdownProcessor = FALSE,
    failed = {},
    store = {};
define

    \* {1, 3}, 0 -> 0
    \* {2, 3}, 0 -> 0
```

CallHistory.tla v11

```
25      \* {1, 2, 3}, 0 -> 3
26      CommitOffset(set, current) ==
27          LET first == current + 1 IN
28          IF /\ first \in set
29             /\ \A i \in {j \in set: j > first}: i-1 \in set
30          THEN CHOOSE i \in set: i+1 \notin set
31          ELSE current
32
33      AllCallsProcessed == <>[](Calls \ failed = store)
34
35      Success == AllCallsProcessed
36
37  end define;
38
39  fair process Call \in Calls
40  begin CallSetup:
41      queue := Append(queue, [id |-> self, action |-> "setup"]);
42  CallComplete:
43      either
44          queue := Append(queue, [id |-> self, action |-> "complete"]);
45      or
46          skip;
47      end either;
48  end process;
```

CallHistory.tla v11

```
49
50  fair process Shutdown = "Shutdown"
51  begin SignalShutdown:
52      when \A c \in Calls: pc[c] = "Done";
53      shutdownProcessor := TRUE;
54  end process;
55
56  fair process Processor = "Processor"
57  variables
58      wip = <<>>,
59      processed = {},
60      crashes = 0,
61      now = 0,
62      offset = commited + 1;
63  begin Process:
64      while TRUE do
65          now := now + 1;
66          either
67              when Len(queue) >= offset;
68  ProcessCDR:
69              with cdr = queue[offset]
70              do
71                  if cdr.id \notin DOMAIN wip /\ cdr.action = "setup" then
72                      wip := Add(wip, cdr.id, [start |-> now, offset |-> offset]);
```

CallHistory.tla v11

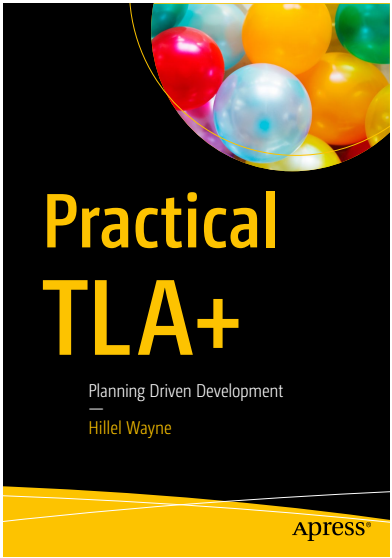
```
73                  elsif cdr.id \in DOMAIN wip /\ cdr.action = "complete" then
74                      processed := processed ++ wip[cdr.id].offset ++ offset;
75                      wip := Remove(wip, cdr.id);
76                      commited := CommitOffset(processed, commited);
77                      store := store ++ cdr.id;
78                  end if;
79              end with;
80              offset := offset + 1;
81          or
82              when shutdownProcessor /\ wip = <<>> /\ offset > Len(queue);
83              goto Done;
84          or
85              when /\ shutdownProcessor
86                   /\ offset > Len(queue)
87                   /\ wip /= <<>>
88                   /\ \A id \in DOMAIN wip: (now - wip[id].start) < Timeout;
89              skip;
90          or
91              when \E id \in DOMAIN wip: (now - wip[id].start) >= Timeout;
92  Cleanup:
93              with
94                  id = CHOOSE id \in DOMAIN wip: (now - wip[id].start) >= Timeout,
95                  off = wip[id].offset
96              do
```

CallHistory.tla v11

```
97                  processed := processed ++ off;
98                  wip := Remove(wip, id);
99                  commited := CommitOffset(processed, commited);
100                 failed := failed ++ id;
101             end with;
102         or
103             when crashes < Crashes;
104 Crash:
105             crashes := crashes + 1 ||
106             processed := {} ||
107             offset := commited + 1 ||
108             wip := <<>>;
109         end either;
110     end while;
111 end process;
112
113 end algorithm; *)
```

©The Cartoon Network

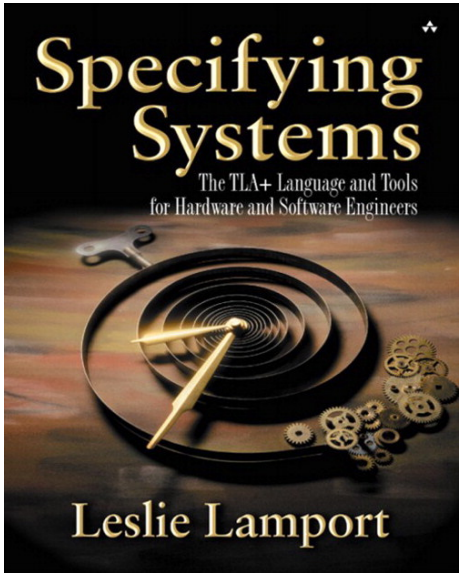
С чего начать?



Practical TLA+
Planning Driven Development

by Hillel Wayne
Apress, 2018

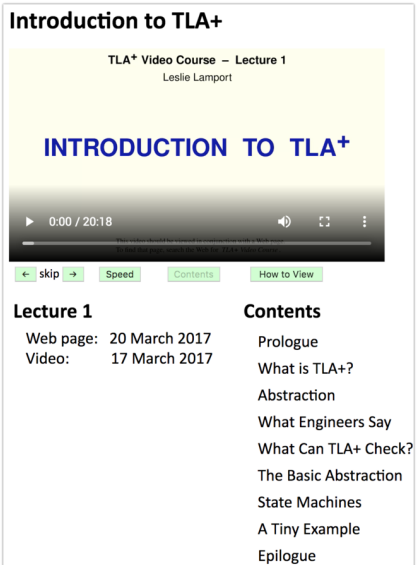
Только про PlusCal, идеально для вхождения в тему!



Specifying Systems
The TLA⁺ Language and Tools for Hardware and Software Engineers

by Leslie Lamport
Addison-Wesley Professional, 2002

TLA⁺, обязательно иметь под рукой



TLA⁺ Video Course

by Leslie Lamport

3.5 часа TLA⁺,

обязательно посмотреть!

(Лэмпорт прекрасный лектор!)

Lamport TLA+ Site: https://lamport.azurewebsites.net/tla/tla.html

Learn TLA+: https://learntla.com

google group: https://groups.google.com/forum/#!forum/tlaplus

reddit: https://www.reddit.com/r/tlaplus/

TLA+ Tools: https://github.com/tlaplus/tlaplus

Q&A

- TLA+ – это недели проектирования и месяцы кодирования?
  - Нет, это часы моделирования и дни или недели работы.
- Что насчет COQ (Agda, Idris, …)?
  - Принципиально разная сложность
  - (есть еще Apalache (`github.com/informalsystems/apalache`) by Informal Systems)
- Что насчет Alloy?
  - Тоже отличный выбор для инженерных задач
- Можно ли из спецификации сразу делать код?
  - Можно, но не нужно (см. PGo (`github.com/UBC-NSS/pgo`))
  - (но если очень нужно, то есть Agda и Idris)

Спасибо!