Interactive TLA+

A. Jesse Jiryu Davis and Samyukta Lanka
MongoDB

Two ways to understand a system

Precise

Does the system obey a particular invariant / property?

Lots of powerful tools for this.

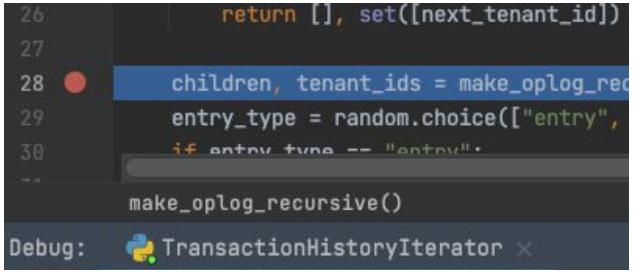
Holistic

Does the system generally conform to my theory of it?

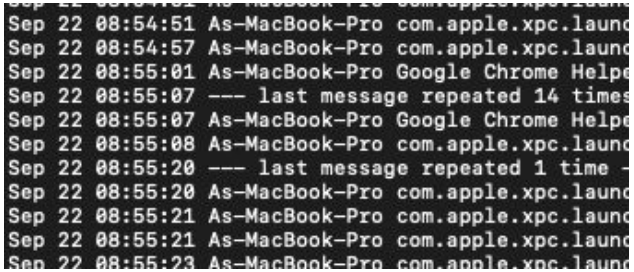
Holistic understanding usually requires interaction or visualization.

Few tools, mostly prototypes.

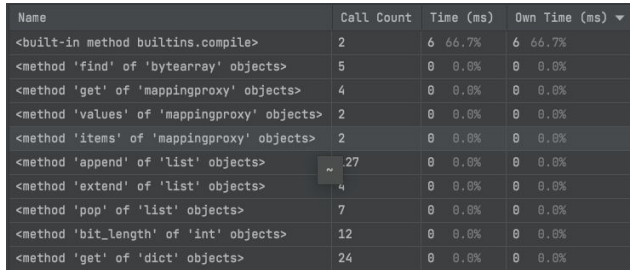
Programmers understand programs holistically through interaction



debugging



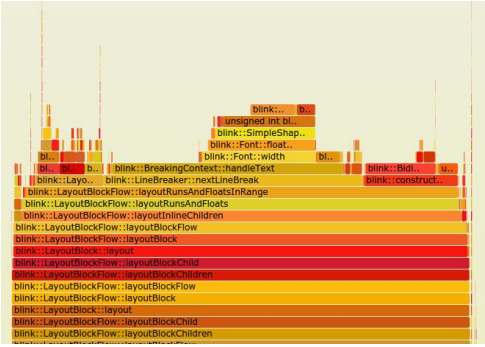
logging



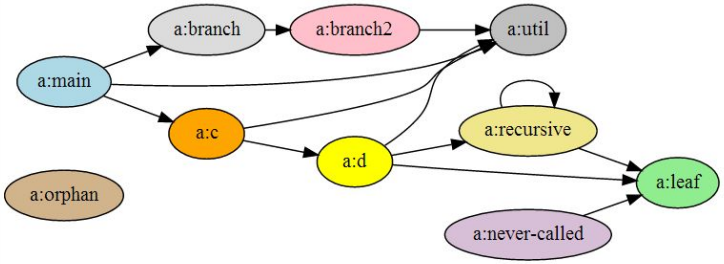
Name	Call Count	Time (ms)	Own Time (ms)
<built-in method builtins.compile>	2	6 66.7%	6 66.7%
<method 'find' of 'bytearray' objects>	5	0 0.0%	0 0.0%
<method 'get' of 'mappingproxy' objects>	4	0 0.0%	0 0.0%
<method 'values' of 'mappingproxy' objects>	2	0 0.0%	0 0.0%
<method 'items' of 'mappingproxy' objects>	2	0 0.0%	0 0.0%
<method 'append' of 'list' objects>	27	0 0.0%	0 0.0%
<method 'extend' of 'list' objects>	4	0 0.0%	0 0.0%
<method 'pop' of 'list' objects>	7	0 0.0%	0 0.0%
<method 'bit_length' of 'int' objects>	12	0 0.0%	0 0.0%
<method 'get' of 'dict' objects>	24	0 0.0%	0 0.0%

profiling

...and visualization



flame charts



call graphs

TLA+ feels like math.

Interaction and visualization are less well-developed for TLA+ than for code.

Let's make it more like programming:
interactive, visual.

Our mission

Review existing tools.

Propose ways to make TLA+ easier for programmers via interaction and visualization.

Your mission

Tell us what tools and techniques we overlooked.

Share your ideas.

Spec authors ask different questions at different times

Does my spec imply my invariants / properties?

Why is my invariant / property false?

What does this TLA+ expression mean?

Is the spec generally behaving as intended?

Why isn't my action enabled?

How did a recent edit change how the spec behaves?

How do I use TLA+ to communicate behaviors to other people?

Main purpose of model-checking & proofs

Decreasingly well-supported

Why is my invariant / property false?

You have a wrong hypothesis about your spec.

What precisely is the mismatch?

Specifying Systems §14.5.2 "Debugging a Specification"

Why is my invariant / property false?

Error traces

.out file

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

@!@!@ENDMSG 2264 @!@!@
@!@!@STARTMSG 2217:4 @!@!@
1: <Initial predicate>
/\ alarmHr = 1
/\ hr = 1
/\ alarmOn = FALSE

@!@!@ENDMSG 2217 @!@!@
@!@!@STARTMSG 2217:4 @!@!@
2: <SetAlarm line 13, col 5 to line 16, col 23 of modu
/\ alarmHr = 7
/\ hr = 1
/\ alarmOn = TRUE

@!@!@ENDMSG 2217 @!@!@
@!@!@STARTMSG 2217:4 @!@!@
3: <SetAlarm line 13, col 5 to line 16, col 23 of modu
/\ alarmHr = 2
/\ hr = 1
/\ alarmOn = TRUE
```

TLA+ Toolbox

Error-Trace

Name	Value
<Initial predicate>	State (num = 1)
alarmHr	1
alarmOn	FALSE
hr	1
<SetAlarm line 13,...	State (num = 2)
alarmHr	7
alarmOn	TRUE
hr	1
<SetAlarm line 13,...	State (num = 3)
alarmHr	2
alarmOn	TRUE
hr	1
<Stuttering>	State (num = 4)

VS Code

Errors

Temporal properties were violated.

Error Trace Filter Hide unmodified

1: Initial predicate
- alarmHr 1
- alarmOn FALSE
- hr 1

2: SetAlarm in AlarmClock >>
- alarmHr M 11
- alarmOn M TRUE
- hr 1

3: SetAlarm in AlarmClock >>
- alarmHr M 8
- alarmOn TRUE
- hr 1

4: Stuttering

tla-trace-formatter (Siyuan Zhou)

TLA+ Trace

State 1: <Initial predicate>		
alarmHr	hr	alarmOn
1	1	FALSE
State 2: <SetAlarm line 13, col 5 to line 16, col 23 of module AlarmClock>		
alarmHr	hr	alarmOn
7	1	TRUE
State 3: <SetAlarm line 13, col 5 to line 16, col 23 of module AlarmClock>		
alarmHr	hr	alarmOn
11	1	TRUE
State 4:		
alarmHr	hr	alarmOn

Why is my invariant / property false?

github.com/visualzhou/tla-trace-formatter

TLA+ Trace

State 1: <Initial predicate>		
alarmHr	hr	alarmOn
1	1	FALSE
State 2: <SetAlarm line 13, col 5 to line 16, col 23 of module AlarmClock>		
alarmHr	hr	alarmOn
7	1	TRUE
State 3: <SetAlarm line 13, col 5 to line 16, col 23 of module AlarmClock>		
alarmHr	hr	alarmOn
11	1	TRUE
State 4:		
alarmHr	hr	alarmOn

What does this TLA+ expression mean?

TLC REPL

```
$ java -cp tla2tools.jar tlc2.REPL
```

What does this TLA+ expression mean?

TLC REPL

```
$ java -cp tla2tools.jar tlc2.REPL
Enter a constant-level TLA+ expression.
(tla+)
```

What does this TLA+ expression mean?

TLC REPL

```
$ java -cp tla2tools.jar tlc2.REPL
Enter a constant-level TLA+ expression.
(tla+) SetToBag({"a", "b"})
```

What does this TLA+ expression mean?

TLC REPL

```
$ java -cp tla2tools.jar tlc2.REPL
Enter a constant-level TLA+ expression.
(tla+) SetToBag({"a", "b"})
[a |-> 1, b |-> 1]
```

What does this TLA+ expression mean?

TLC REPL

```
$ java -cp tla2tools.jar tlc2.REPL
Enter a constant-level TLA+ expression.
(tla+) SetToBag({"a", "b"})
[a |-> 1, b |-> 1]
(tla+) SetToBag({1, 2})
```

What does this TLA+ expression mean?

TLC REPL

```
$ java -cp tla2tools.jar tlc2.REPL
Enter a constant-level TLA+ expression.
(tla+) SetToBag({"a", "b"})
[a |-> 1, b |-> 1]
(tla+) SetToBag({1, 2})
<<1, 1>>
```

Is the spec behaving as intended?

Print() expressions are confusing in model-checking mode

```
---------------------- MODULE HourClock ---------------------
EXTENDS Naturals, TLC
VARIABLE hr
HCini == hr \in (1 .. 12)
HCnxt ==
    /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
    /\ PrintT(<<"hr is ", hr, "hr' is", hr'>>)
HC == HCini /\ [][HCnxt]_hr
=============================================================
```

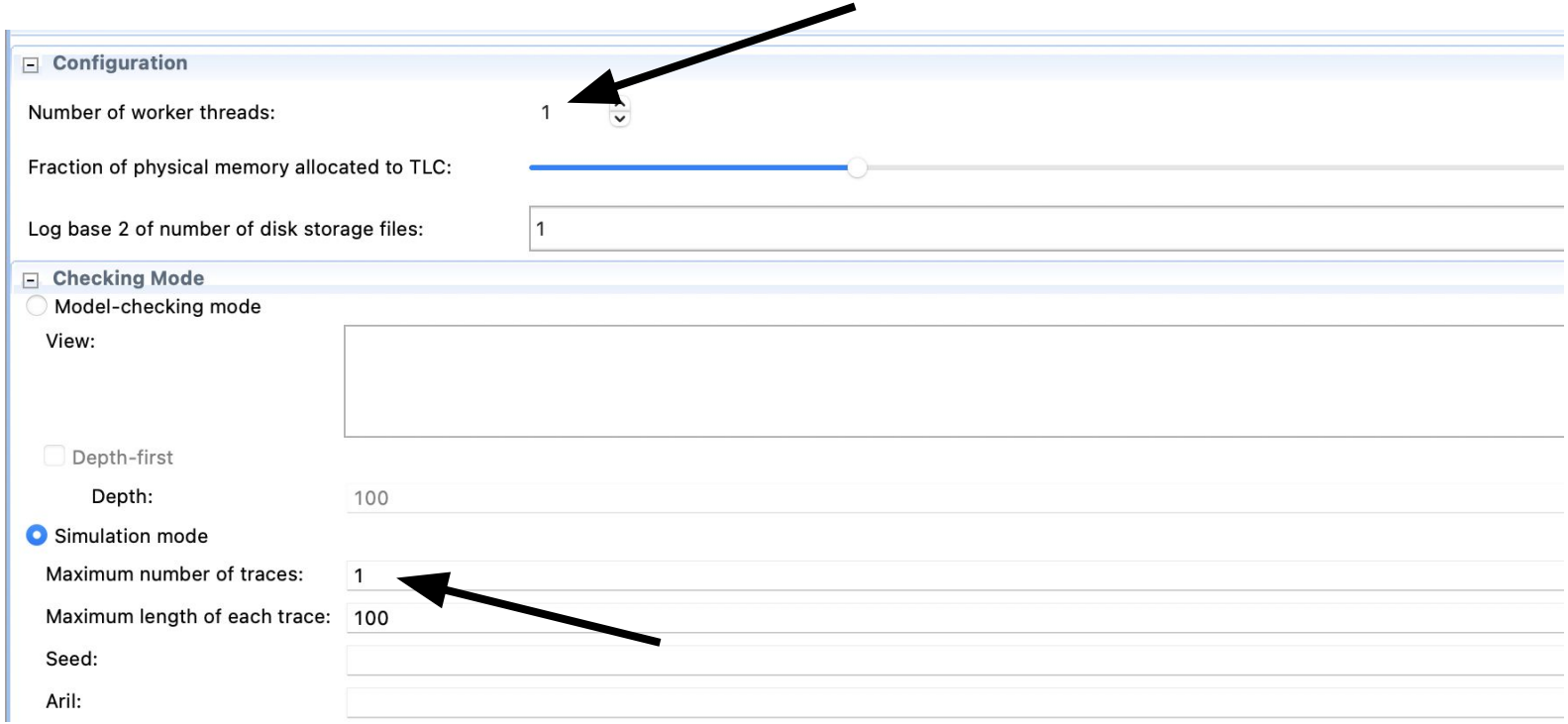
Is the spec behaving as intended?

Print() expressions are confusing in model-checking mode

```
<<"hr is ", 4, "hr' is", 5>>
<<"hr is ", 3, "hr' is", 4>>
<<"hr is ", 2, "hr' is", 3>>
<<"hr is ", 5, "hr' is", 6>>
<<"hr is ", 1, "hr' is", 2>>
<<"hr is ", 6, "hr' is", 7>>
<<"hr is ", 10, "hr' is", 11>>
<<"hr is ", 8, "hr' is", 9>>
<<"hr is ", 12, "hr' is", 1>>
<<"hr is ", 11, "hr' is", 12>>
<<"hr is ", 9, "hr' is", 10>>
<<"hr is ", 7, "hr' is", 8>>
```

Is the spec behaving as intended?

Print() expressions plus **simulation mode**



Is the spec behaving as intended?

Print() expressions plus **simulation mode**

```
<<"hr is ", 4, "hr' is", 5>>
<<"hr is ", 5, "hr' is", 6>>
<<"hr is ", 6, "hr' is", 7>>
<<"hr is ", 7, "hr' is", 8>>
<<"hr is ", 8, "hr' is", 9>>
<<"hr is ", 9, "hr' is", 10>>
<<"hr is ", 10, "hr' is", 11>>
<<"hr is ", 11, "hr' is", 12>>
<<"hr is ", 12, "hr' is", 1>>
<<"hr is ", 1, "hr' is", 2>>
<<"hr is ", 2, "hr' is", 3>>
<<"hr is ", 3, "hr' is", 4>>
<<"hr is ", 4, "hr' is", 5>>
<<"hr is ", 5, "hr' is", 6>>
<<"hr is ", 6, "hr' is", 7>>
```

Is the spec behaving as intended?

Simulation mode — constraining the model to show interesting traces

What is the behavior spec?

Initial predicate and next-state relation

Init:

```
hr = 4
```

Next:

```
HCnxt
```

Is the spec behaving as intended?

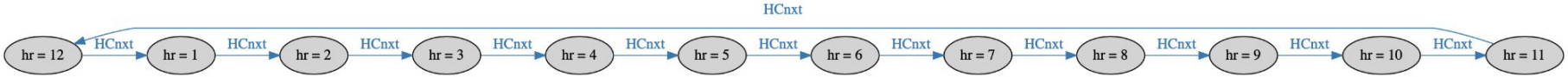
GraphViz

TLC command line parameters:

```
-dump dot,colorize,actionlabels HourClock.dot
```

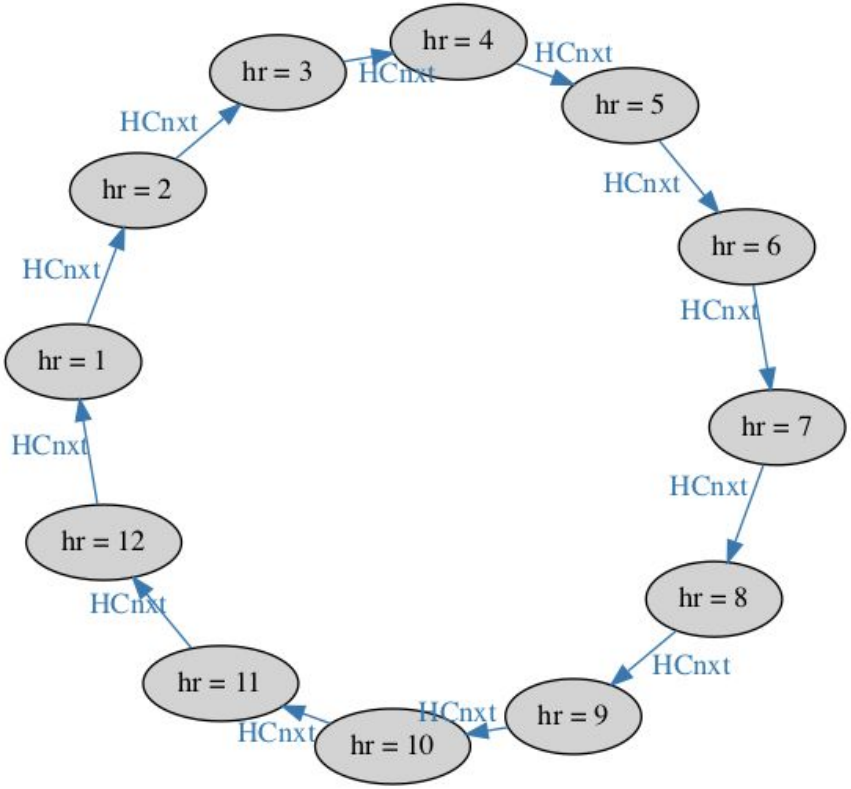
Is the spec behaving as intended?

GraphViz



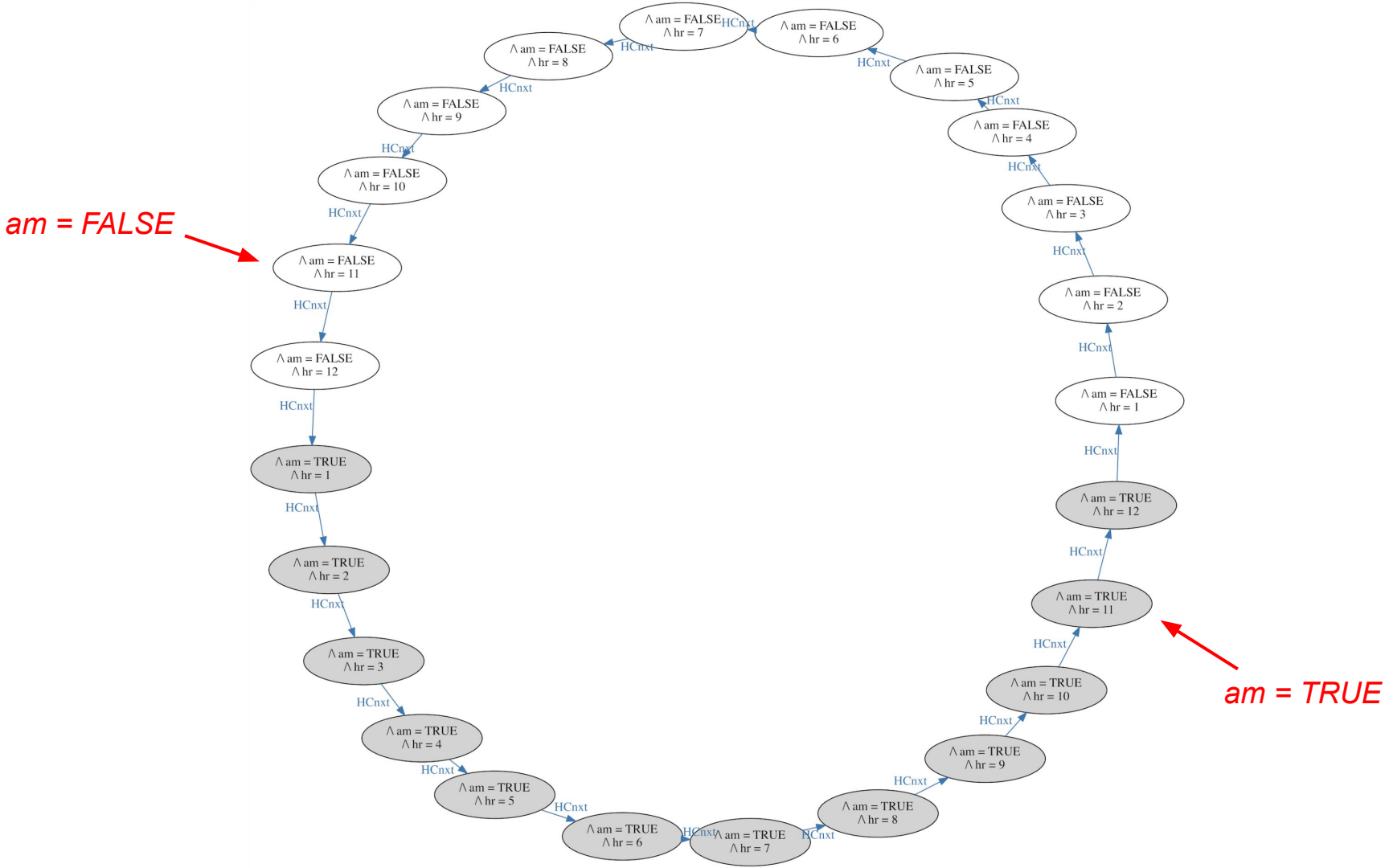
Is the spec behaving as intended?

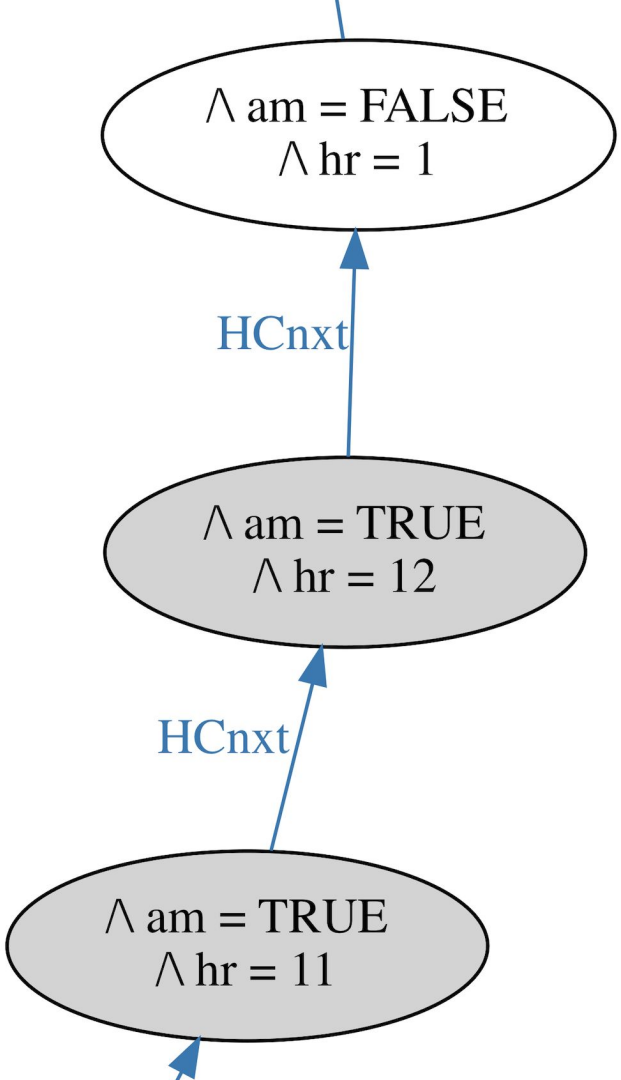
GraphViz

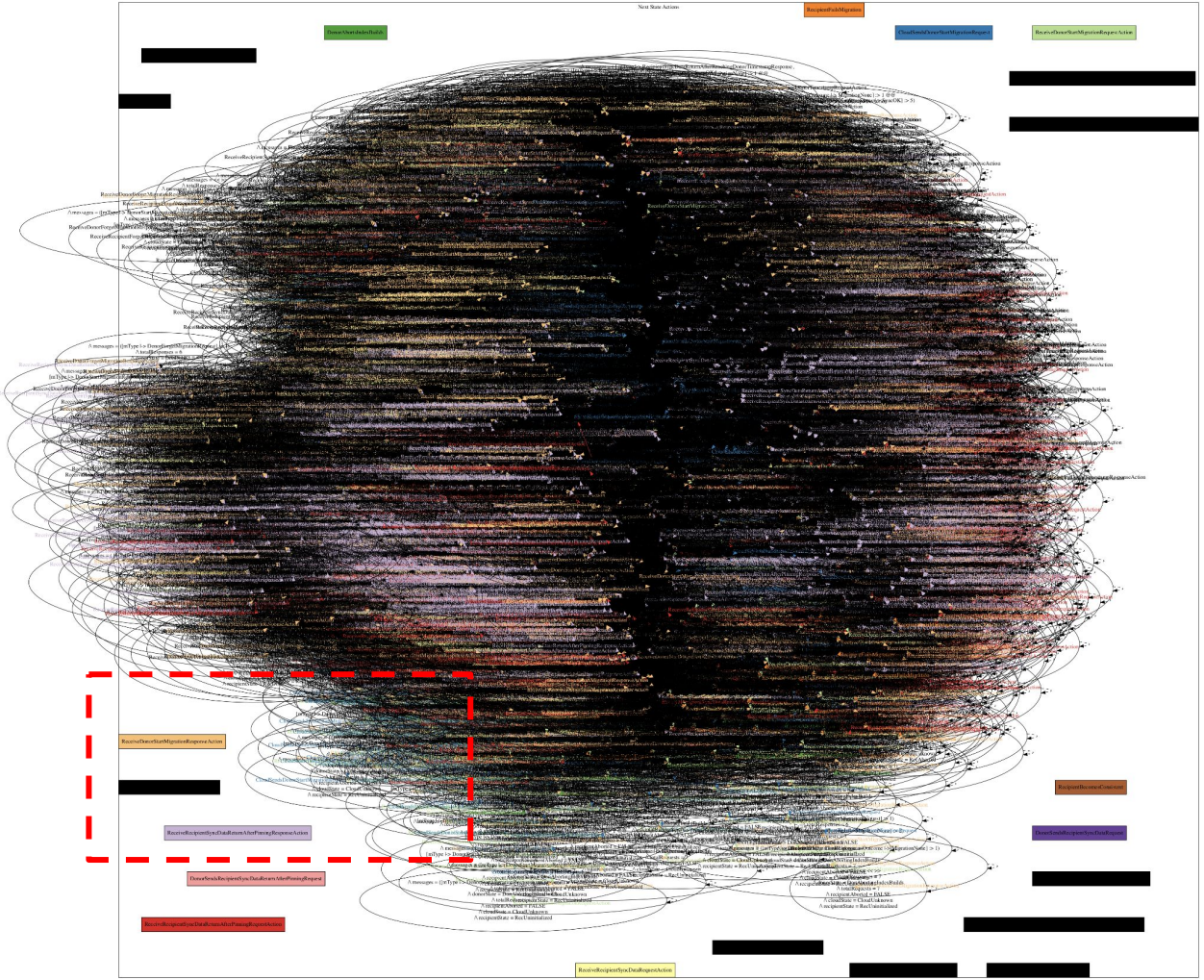


Is the spec behaving as intended?

```
\* Incorrectly add am/pm to HourClock
---------------------- MODULE HourClockAMPM ------------------
EXTENDS Naturals
VARIABLE hr, am
HCini == hr \in (1 .. 12) /\ am = TRUE
HCnxt ==
    /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
    \* Oops, AM/PM should flip at noon/midnight, not 1 o'clock.
    /\ am' = IF hr = 12 THEN ~am ELSE am
HC == HCini /\ [][HCnxt]_<<hr, am>>
=================================================================
```


am = FALSE
am = TRUE
∧ am = FALSE ∧ hr = 7
∧ am = FALSE ∧ hr = 6
∧ am = FALSE ∧ hr = 8
∧ am = FALSE ∧ hr = 5
∧ am = FALSE ∧ hr = 9
∧ am = FALSE ∧ hr = 4
∧ am = FALSE ∧ hr = 10
∧ am = FALSE ∧ hr = 3
∧ am = FALSE ∧ hr = 11
∧ am = FALSE ∧ hr = 2
∧ am = FALSE ∧ hr = 12
∧ am = FALSE ∧ hr = 1
∧ am = TRUE ∧ hr = 1
∧ am = TRUE ∧ hr = 12
∧ am = TRUE ∧ hr = 2
∧ am = TRUE ∧ hr = 11
∧ am = TRUE ∧ hr = 3
∧ am = TRUE ∧ hr = 10
∧ am = TRUE ∧ hr = 4
∧ am = TRUE ∧ hr = 9
∧ am = TRUE ∧ hr = 5
∧ am = TRUE ∧ hr = 8
∧ am = TRUE ∧ hr = 6
∧ am = TRUE ∧ hr = 7
HCnxt


∧ am = FALSE
∧ hr = 1
HCnxt
∧ am = TRUE
∧ hr = 12
HCnxt
∧ am = TRUE
∧ hr = 11


Next State Actions
RecipientFailedMigration
DonorAbortsIndexBuilds
CloudSendsDonorStartMigrationRequest
ReceiveDonorStartMigrationRequestAction
ReceiveDonorStartMigrationResponseAction
RecipientRecommitCommitment
ReceiveRecipientSyncDataReturnAfterPruningResponseAction
DonorSendsRecipientSyncDataReturnAfterPruningRequest
ReceiveRecipientSyncDataReturnAfterPruningRequestAction
DonorSendsRecipientSyncDataRequest
ReceiveRecipientSyncDataRequestAction

ReceiveDonorStartMigrationResponseAction

[redacted]

black is the new black

ReceiveRecipientSyncDataReturnAfterPinningResponseAction

Is the spec behaving as intended?

Profiling

New example: an alarm clock.

```
VARIABLES hr, alarmHr, alarmOn
vars == <<hr, alarmHr, alarmOn>>
HCini ==
    /\ hr \in (1 .. 12)
    /\ alarmHr \in (1..12)
    /\ alarmOn = FALSE
AdvanceHour ==
    /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
    /\ UNCHANGED <<alarmHr, alarmOn>>
SetAlarm ==
    /\ alarmHr' \in (1..12)
    \* Oops, forgot to set alarmOn' = TRUE
    /\ UNCHANGED <<hr, alarmOn>>
Ring ==
    /\ alarmOn
    /\ hr = alarmHr
    /\ alarmOn' = FALSE
    /\ UNCHANGED <<alarmHr, hr>>
HC == HCini /\ [][AdvanceHour \/ SetAlarm \/ Ring]_vars /\ SF_vars(Ring)
```

oops, alarmOn is always FALSE

Is the spec behaving as intended?

Profiling

Module	Action	Location	States Found	Distinct States
AlarmClock	AdvanceHour	line 9, col 1 to line 9, col 11	144	0
AlarmClock	SetAlarm	line 12, col 1 to line 12, col 8	1,728	0
AlarmClock	Ring	line 16, col 1 to line 16, col 4	0	0
AlarmClock	HCini	line 5, col 1 to line 5, col 5	144	144

Is the spec behaving as intended?

Profiling

```
1  ---------------------- MODULE AlarmClock ----------------------
2  EXTENDS Naturals
3  VARIABLES hr, alarmHr, alarmOn
4  vars == <<hr, alarmHr, alarmOn>>
5  HCini ==
6      /\ hr \in (1 .. 12)
7      /\ alarmHr \in (1..12)
8      /\ alarmOn = FALSE
9  AdvanceHour ==
10     /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
11     /\ UNCHANGED <<alarmHr, alarmOn>>
12 SetAlarm ==
13     /\ alarmHr' \in (1..12)
14     \* Oops, forgot to set alarmOn' = TRUE
15     /\ UNCHANGED <<hr, alarmOn>>
16 IRing is never enabled.
17     /\ alarmOn
18     /\ hr = alarmHr
19     /\ alarmOn' = FALSE
20     /\ UNCHANGED <<alarmHr, hr>>
21 HC == HCini /\ [][AdvanceHour \/ SetAlarm \/ Ring]_vars /\ SF_vars(Ring)
22 ==================================================================
```

Is the spec behaving as intended?

Profiling

```
1  ---------------------- MODULE AlarmClock ----------------------
2  EXTENDS Naturals
3  VARIABLES hr, alarmHr, alarmOn
4  vars == <<hr, alarmHr, alarmOn>>
5  HCini ==
6      /\ hr \in (1 .. 12)
7      /\ alarmHr \in (1..12)
8      /\ alarmOn = FALSE
9  AdvanceHour ==
10     /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
11     /\ UNCHANGED <<alarmHr, alarmOn>>
12 SetAlarm ==
13     /\ alarmHr' \in (1..12)
14     \* Oops, forgot to set alarmOn' = TRUE
15     /\ UNCHANGED <<hr, alarmOn>>
16 Ring ==
17     /\ alarmOn
18     /\ hr = alarmHr
19     /\ alarmOn' = FALSE
20     /\ UNCHANGED <<alarmHr, hr>>
21 HC == HCini /\ [][AdvanceHour \/ SetAlarm \/ Ring]_vars /\ SF_vars(Ring)
22 ==================================================================
```

Feature proposal: fail model-checking if any action is never enabled

uh oh

Why isn't my action enabled?

"Staring really hard"?

This is an area for research.

Why isn't my action enabled?

```
Push(stack, x) ==
    stack' = Append(stack, x)

Pop(stack) ==
    stack' = SubSeq(stack, 1, Len(stack) - 1)

Init == myStack = <<"x">>

SomeAction ==
    /\ Pop(myStack)
    /\ Push(myStack, "y")
```

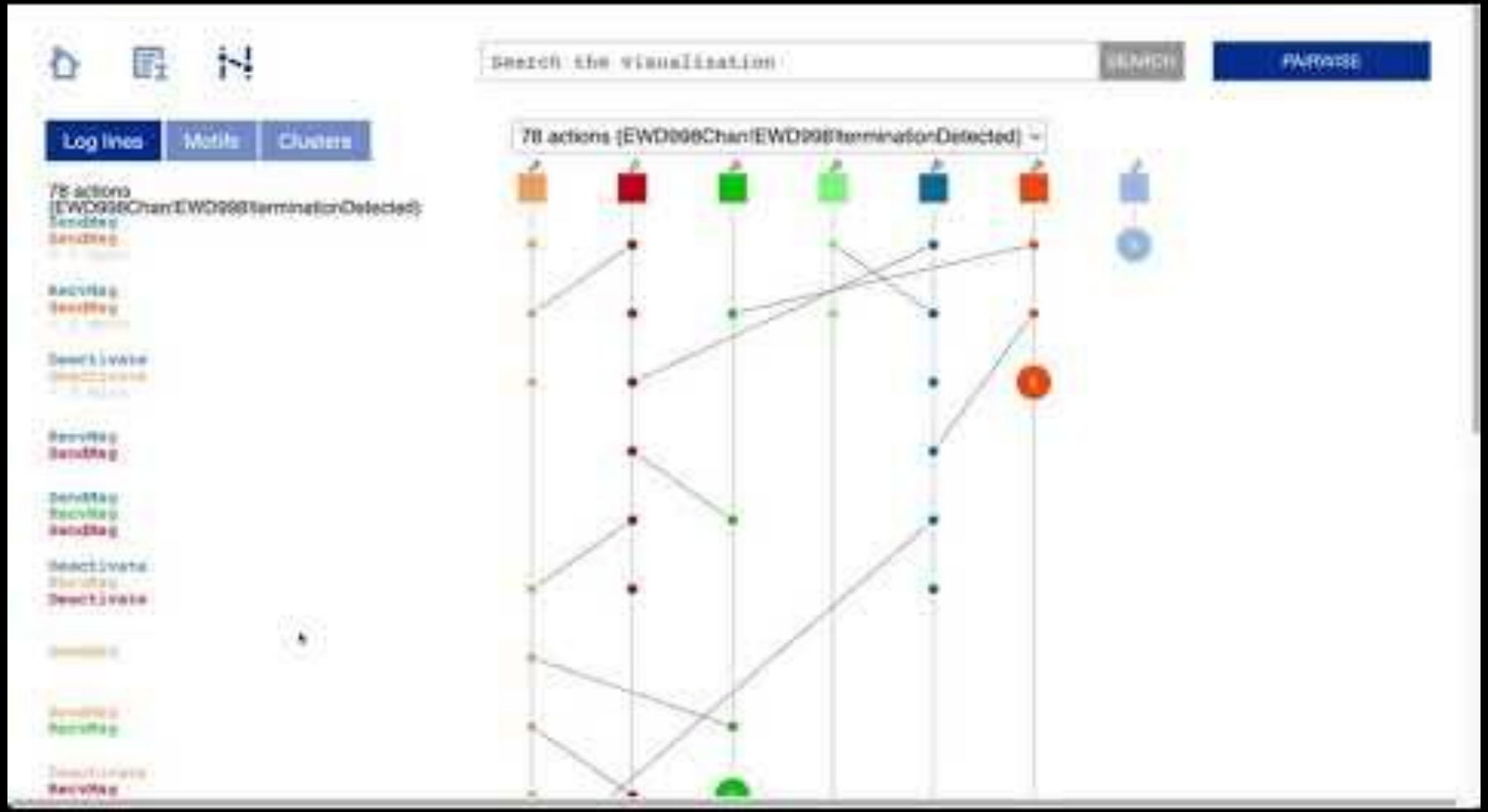
equivalent to stack = <<>> /\ stack = <<"y">>
which is FALSE

Proposal: prohibit contradictory uses
of a primed variable in an action

Is the spec behaving as intended?

ShiViz

For specs with multiple processes that exchange messages with a vector clock


Search the visualisation
SEARCH
PARSE
Log lines
Motifs
Clusters
78 actions (EWD998Chan!EWD998!terminationDetected)
78 actions (EWD998Chan!EWD998!terminationDetected)

```
------------ MODULE EWD998ChanID_shiviz -------------------
EXTENDS EWD998ChanID, Json

(* ... deleted code ... *)

Alias ==
    [
        Host |-> host
        ,Clock |-> ToJsonObject(clock[host])

        ,active |-> active
        ,color |-> color
        ,counter |-> counter
        ,inbox |-> inbox
    ]
=================================================================
```

#motif

< 10 INSTANCES IN VIEW >

Log lines | Motifs | Clusters

Find network motifs:

- [ ] 2-event motifs
- [x] 3-event motifs
- [ ] 4-event motifs

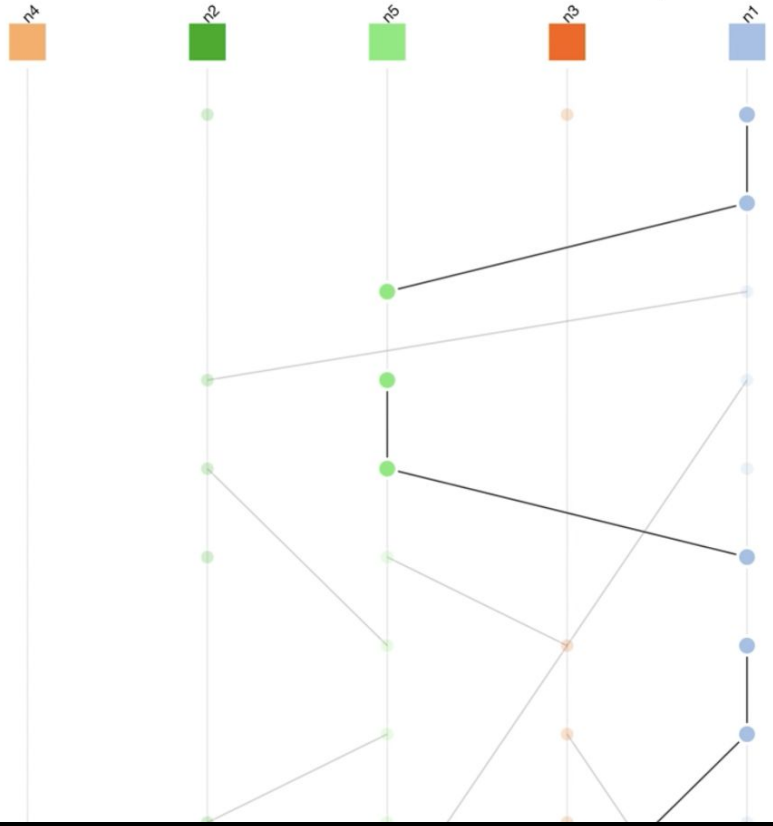
Motif 1 :

666 actions: 24 instances
▸ 249 actions: 10 instances

Motif 2 :

666 actions: 96 instances
249 actions: 37 instances
78 actions (EWD998Chan!EWD998!terminationDetected): 10 instances

249 actions



SendMsg

host: n3
active: (n1 :> FALSE @@ n2 :> TRUE @@ n3 :> TRUE @@ n4 :> TRUE @@ n5 :> TRUE @@ n6 :> TRUE @@ n7 :> TRUE)

How did a recent edit change how the spec behaves?

```
\* Incorrectly add am/pm to HourClock
---------------------- MODULE HourClockAMPM ------------------
EXTENDS Naturals
VARIABLE hr, am
HCini == hr \in (1 .. 12) /\ am = TRUE
HCnxt ==
    /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
    \* Oops, AM/PM should flip at noon/midnight, not 1 o'clock.
    /\ am' = IF hr = 12 THEN ~am ELSE am
HC == HCini /\ [][HCnxt]_<<hr, am>>
=============================================================
```

How did a recent edit change how the spec behaves?

TLA+ Debugger

RUN AND DEBUG

VARIABLES

Action

HCnxt: [am |-> TRUE, am' |-> ?, hr |-> 2, hr' |-> 3…

am : TRUE

am': ?

hr : 2

hr': 3

Trace

2: <HCnxt line 7, col 5 to line 8, col 40 of module …

am: ?

hr: 3

1: <HCini line 5, col 10 to line 5, col 38 of module…

am: TRUE

hr: 2

HourClockAMPM.tla

Users > samy > Documents > HourClockAMPM.tla > {} HourClockAMPM

```
1   ---------------------------------- MODULE HourClockAMPM --
2   EXTENDS Naturals, TLC
3   VARIABLES hr, am
4
5   HCini == hr \in (1 .. 12) /\ am = TRUE
6   HCnxt ==
7       /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
8       /\ am' = IF hr = 12 THEN ~am ELSE am
9   HC == HCini /\ [][HCnxt]_<<hr, am>>
10
11  ==========================================================
12
```

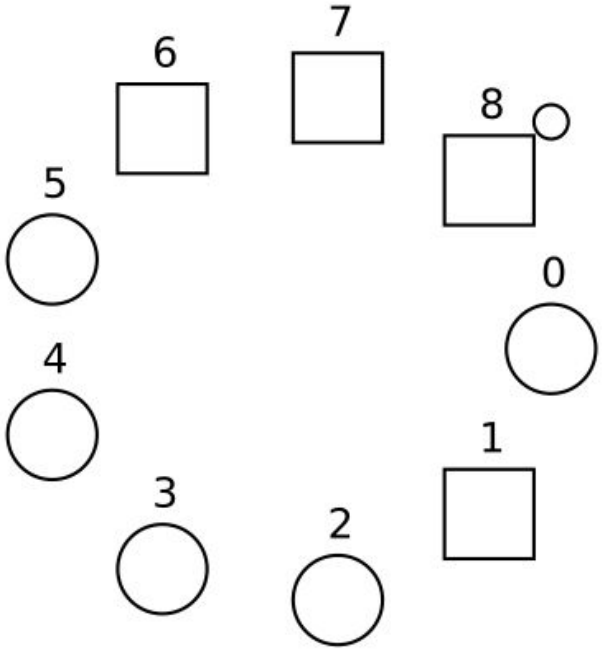
How do I use TLA+ to communicate behaviors to other people?

TLA+ Animation: https://github.com/will62794/tlaplus_animation

Circle: Active, Black: Tainted

Line: Message, Arrow: Receiver

Level: 1



Future of interactive TLA+ spec development

Tools already exist to address precise questions

Let’s build more tools to help us better holistically understand specs

Iterative Spec Development

TLA+ Debugger is one way to achieve this

Key: Quickly see effects of our changes

Extensions to the TLA+ Debugger: Watchpoints

HourClockAMPM.tla

Users > samy > Documents > HourClockAMPM.tla > ...

```
\* Incorrectly add am/pm to HourClock
---------------------------------- MODULE HourClockAMPM ----------
EXTENDS Naturals, TLC
VARIABLES hr, am

HCini == hr \in (1 .. 12) /\ am = TRUE
HCnxt ==
    /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
    \* Oops, AM/PM should flip at noon/midnight, not 1 o'clock.
    /\ am' = IF hr = 12 THEN ~am ELSE am
HC == HCini /\ [][HCnxt]_<<hr, am>>

==========================================================================
```

Extensions to the TLA+ Debugger: Watchpoints

RUN AND DEBUG

VARIABLES

Action

HCnxt: [am |-> TRUE, am' |-> FALSE, hr |-> 12, hr' …

am : TRUE

am': FALSE

hr : 12

hr': 1

Trace

2: <HCnxt line 8, col 5 to line 10, col 40 of module…

am: FALSE

hr: 1

1: <HCini line 6, col 10 to line 6, col 38 of module…

am: TRUE

hr: 12

HourClockAMPM.tla

Users > samy > Documents > HourClockAMPM.tla > { } HourClockAMPM > H

```
1   \* Incorrectly add am/pm to HourClock
2   ---------------------------------- MODULE HourClockAMPM -----
3   EXTENDS Naturals, TLC
4   VARIABLES hr, am
5
6   HCini == hr \in (1 .. 12) /\ am = TRUE
7   HCnxt ==
8       /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
9       \* Oops, AM/PM should flip at noon/midnight, not 1 o'clock.
10      /\ am' = IF hr = 12 THEN ~am ELSE am
11  HC == HCini /\ [][HCnxt]_<<hr, am>>
12
13  ==================================================================
14
```

Extensions to the TLA+ Debugger: Conditional Breakpoints

Breakpoint that only pauses execution if the supplied predicate is true

```
hr = 12

hr \in {11, 12, 1}

am = false
```

Extensions to the TLA+ Debugger: Conditional Breakpoints

RUN AND DEBUG

VARIABLES

Action

HCnxt: [am |-> TRUE, am' |-> ?, hr |-> 12, hr' |-> …

am : TRUE

am': ?

hr : 12

hr': 1

Trace

2: <HCnxt line 8, col 5 to line 10, col 40 of module…

am: ?

hr: 1

1: <HCini line 6, col 10 to line 6, col 38 of module…

am: TRUE

hr: 12

HourClockAMPM.tla

Users > samy > Documents > HourClockAMPM.tla > {} HourClockAMPM >

```
1   \* Incorrectly add am/pm to HourClock
2   ---------------------------------- MODULE HourClockAMPM -
3   EXTENDS Naturals, TLC
4   VARIABLES hr, am
5
6   HCini == hr \in (1 .. 12) /\ am = TRUE
7   HCnxt ==
8       /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
9       \* Oops, AM/PM should flip at noon/midnight, not 1 o'clo
10      /\ am' = IF hr = 12 THEN ~am ELSE am
11  HC == HCini /\ [][HCnxt]_<<hr, am>>
12
13  ==========================================================
14
```

Iterative Spec Development

TLA+ Debugger is one way to achieve this

Key: Quickly see effects of our changes

Key: Experiment with expressions

Extensions to the TLA+ Debugger: Watch Expressions

RUN AND DEBUG

VARIABLES

WATCH

Expression to watch

HourClockAMPM.tla

Users > samy > Documents > HourClockAMPM.tla > ...

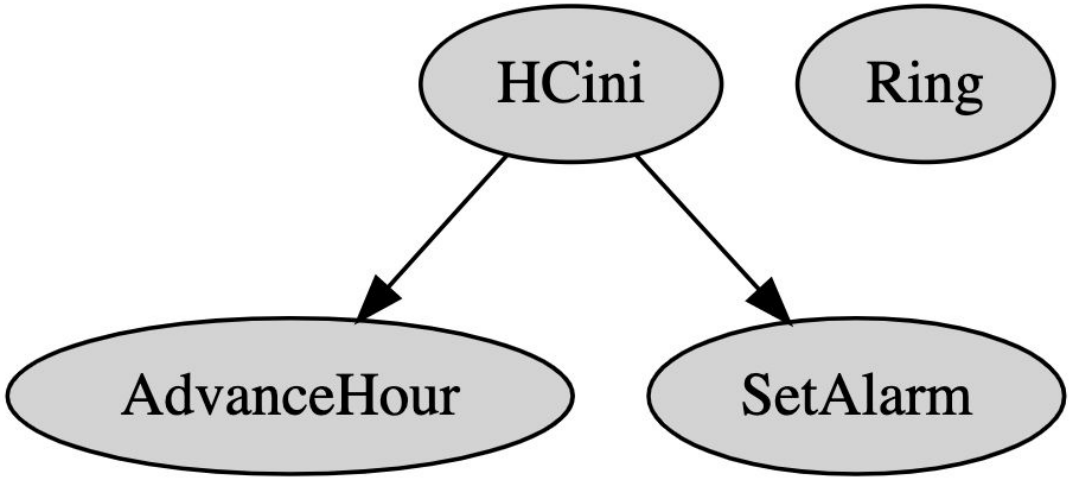
```
1  ---------------------------------- MODULE HourClockAMPM ----------------
2  EXTENDS Naturals, TLC
3  VARIABLES hr, am
4
5  HCini == hr \in (1 .. 12) /\ am = TRUE
6  HCnxt ==
7      /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
8      /\ am' = IF hr = 12 THEN ~am ELSE am
9  HC == HCini /\ [][HCnxt]_<<hr, am>>
10
11 ==========================================================================
12
```

Iterative Spec Development

Should be able to better understand if our spec behaves as intended

Still doesn’t take into account areas of the spec we *didn’t* inspect

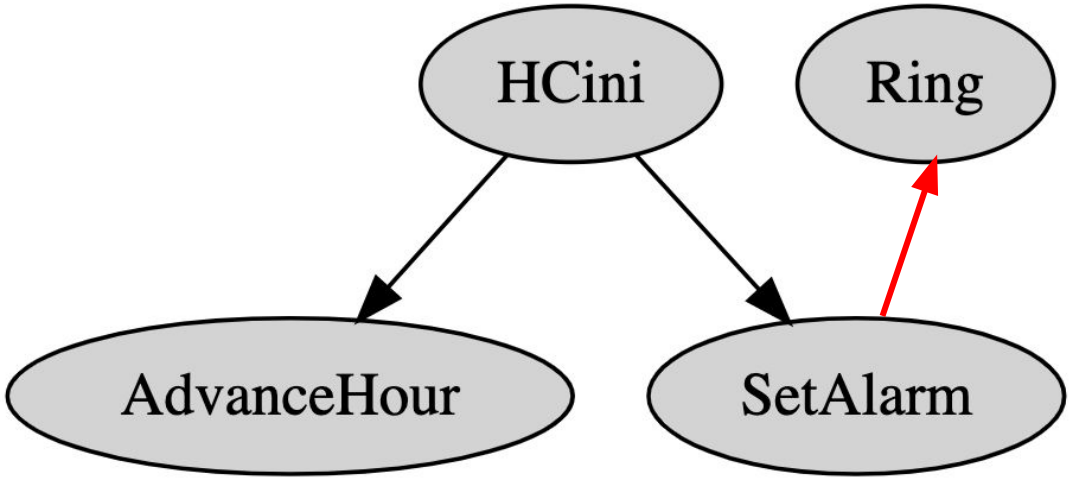
Graph of actions that enable other actions



```
VARIABLES hr, alarmHr, alarmOn
vars == <<hr, alarmHr, alarmOn>>
HCini ==
    /\ hr \in (1 .. 12)
    /\ alarmHr \in (1..12)
    /\ alarmOn = FALSE
AdvanceHour ==
    /\ hr' = IF hr # 12 THEN hr + 1 ELSE 1
    /\ UNCHANGED <<alarmHr, alarmOn>>
SetAlarm ==
    /\ alarmHr' \in (1..12)
    \* Oops, forgot to set alarmOn' = TRUE
    /\ UNCHANGED <<hr, alarmOn>>
Ring ==
    /\ alarmOn
    /\ hr = alarmHr
    /\ alarmOn' = FALSE
    /\ UNCHANGED <<alarmHr, hr>>
HC == HCini /\ [][AdvanceHour \/ SetAlarm \/ Ring]_vars /\ SF_vars(Ring)
```

oops, alarmOn is always FALSE

Graph of actions that enable other actions



“Always Be Suspicious of Success”

But where do we direct our suspicions?

We need more sanity checks that don’t rely on us defining the perfect invariant

Sanity Check: Variable Ranges

```
---------------------- MODULE Loop -----------------
EXTENDS Naturals
VARIABLE x
Init == x \in (1 .. 10)
ActionOne ==
   /\ x = 10
   /\ x' = 1
ActionTwo ==
   \* Oops, this could cause x' to be 11.
   /\ x' = x + 1
...
=================================================================
```

x \in 1..100

```
---------------------- MODULE Loop ------------------
EXTENDS Naturals
VARIABLE x
Init == x \in BOOLEAN
Action ==
    /\ x
   \* Oops, we meant x' = ~x
    /\ x' = x
...
==========================================================
```

Values of x:
TRUE: 95%
FALSE: 5%

Questions for the Audience

What use cases did we miss? What questions have you had about a spec that you didn’t know how to answer?

What features and tools did we miss? How can they be better promoted so programmers like us would find them next time?

What's the right direction for making TLA+ easier - more tools, or consolidate more features in one tool? Is that one tool the Toolbox or VS Code or what?