Joshua

LosAngeles971, 2021

Joshua

> I am not afraid of how much a machine can look like a human being, but how much a human being can be assimilated to a complex machine.
>
> — @Bluepulsar971

Joshua* is a software. It was born from an inspiration coming from reading a book [1], and its purpose is solving generics problems using a combination of knowledge **and** state**. The knowledge consists of** events **and relationships between events. An event is *something* that can occur under certain circumstances, for example an event may be "the farmer reaches the est bank of the river", and it can occur if the farmer is on the ovest bank of the river. A relationship ties two events under the principle of** cause-effect**, thus the occurrence of an event can cause the occurrence of other events. Finally, the state is a set of** variables **which can determine the conditions that allow an event to happen, as well as the occurrence of every event can change the state. How can Joshua solve a problem? Given a base knowledge, a problem can be described as the combination of an initial state and an event, and finding the solution means answering the question:** is there at leat one chain of events that eventually cause my event?**.

[1] Massimo Chiriatti - ???

1. Introduction

TBD

1.1. The farmer, the wolf, the goat and the cabbage

> Once upon a time a farmer went to a market and purchased a wolf, a goat, and a cabbage. On his way home, the farmer came to the bank of a river and rented a boat. But crossing the river by boat, the farmer could carry only himself and a single one of his purchases: the wolf, the goat, or the cabbage. If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage. The farmer's challenge was to carry himself and his purchases to the far bank of the river, leaving each purchase intact. How did he do it?
>
> — https://en.wikipedia.org/wiki/Wolf, _goat_and_cabbage_problem

Joshua describes a problem as the combination of an initial **state** and a desired **effect**, and finding the solution means answering the question: is there at least one **chain of concatanted events** that eventually cause the desired effect?**

An event is something that can occur, like a phisycal phenomena or a statement, and it can occur under certain *circumstances* coming from the current **state**. For example, "The farmer brings the cabbage on the bank B of the river" can occur if both farmer and cabbage are on the bank A of the river. The state is a set of **variables** that determine the **conditions** that allow an event to happen, as well as the occurrence of an event change the current state. Technically speaking, the definition of an event includes:

- a description
- an *optional* list of conditions
- an *optional* list of assignments
- an *optional* list of effects

An effect is an event that assumes a **cause-effect relationship** with another event, and the relationship is characterized by a weight. Indeed, Joshua recognizes three types of cause-effect relationships depending on the weight:

- a weight of 0 means that if the cause occurs than the effect **never** occurs
- a weight of 1 means that if the cause occurs than the effect **always** occurs
- any weight into (0,1) means if the cause occurs than the effect **may** occurs

For example "The farmer brings the cabbage on the bank B of the river" is the cause of the effect "The farmer, the wolf, the goat and the cabbage are on the bank B of the river", but the weight of their relationship is a value into the interval (0,1) like 0.5, since it is not sure that the cause's occurrence triggers the effect.

Nevetheless, an event occurs under specific circumstances determined by the current state, and the latter is a set of variables. Thinking of Farmer's problem, the state may be defined by 8 variables to determine if a character is on the river's bank A (value 1) or not (value 0), therefore the initial state

may be:

- FarmerA = 1
- WolfA = 1
- GoatA = 1
- CabbageA = 1
- FarmerB = 0
- WolfB = 0
- GoatB = 0
- CabbageB = 0

Now it is possible to take a look of a complete definition of an event.

Table 1. Cause-effect relationships

Event	"The farmer brings the cabbage on the bank B of the river"
Conditions	• FarmerA == 1 • CabbageA == 1 • WolfA < 1) && (GoatA == 1
	WolfA == 1) && (GoatA < 1
Effect	"The farmer, the wolf, the goat and the cabbage are on the bank B of the river" with weight 0.5
Assignments	*FarmerB = 1 * CabbageB = 1 * FarmerA = 0 * CabbageA = 0

The **assignments** express the consequences of the event's occurrence (not the effect). If the "The farmer brings the cabbage on the bank B of the river" occur, than state must change accordingly.

A set of well known events and cause-effect relationships forms the base knowledge used by Joshua to solve a **problem**. Even if solving problems may increase the knowledge, Joshua is not able (or not deisgned to) to build up the knonledge from scratch, since it requires the ability to observe and understand the occurrence of not known events, in order to classify them and deriving new cause-effect relationships.

In simple workds, **solving a problem** is understanding if an event can happen. Solving the Farmer's problem means understanding what conditions make the event "The farmer, the wolf, the goat and the cabbage are on the bank B of the river" occurs, and such conditions are a chain of events.

NOTE	Joshua defines a solution as a concatenation of events, linked together by the cause-effect principle, in such a way that the first event triggers a succession of events that eventually leads to the occurrence of the desired effect.

Being able to describe a problem is everything, nevertheless it can be tricky, since the description has to be limited to declare the facts (events, relationships and state), without falling into the desire of anticipating the reasoning.

Describing a problems means asking the questions:

1. Which event represents the solution?
2. Which others events are possible?
3. Which cause-effect relationships ties the events together?

Here the events related to the Farmer's problem:

- The farmer, the wolf, the goat and the cabbage are on the bank B of the river (solution)
- The farmer brings the cabbage on the bank B of the river
- The farmer brings the cabbage on the bank A of the river
- The farmer brings the goat on the bank B of the river
- The farmer brings the goat on the bank A of the river
- The farmer brings the wolf on the bank B of the river
- The farmer brings the wolf on the bank A of the river
- The farmer goes to the bank A of the river
- The farmer goes to the bank B of the river

Of course, every event must be defined entirely, in terms of conditions, effects and assignements.

Here how Joshua defines the base knowledge to solve the Farmer's problem, using a YAML [2] file.

```
  knowlegde:
    - id: "The farmer, the wolf, the goat and the cabbage are on the bank B of the
river"
      conditions:
        - FarmerB == 1
        - WolfB == 1
        - GoatB == 1
        - CabbageB == 1

    - id: The farmer brings the cabbage on the bank B of the river
      conditions:
        - FarmerA == 1
        - CabbageA == 1
        - ((WolfA < 1) && (GoatA == 1)) || ((WolfA == 1) && (GoatA < 1))
      effects:
        - weight: 0.5
```

```
          event: "The farmer, the wolf, the goat and the cabbage are on the bank B of
the river"
      assignments:
        - FarmerB = 1
        - CabbageB = 1
        - FarmerA = 0
        - CabbageA = 0

    - id: "The farmer brings the cabbage on the bank A of the river"
      conditions:
        - FarmerB == 1
        - CabbageB == 1
        - ((WolfB < 1) && (GoatB == 1)) || ((WolfB == 1) && (GoatB < 1))
      effects:
        - weight: 0.1
          event: "The farmer, the wolf, the goat and the cabbage are on the bank B of
the river"
      assignments:
        - FarmerA = 1
        - CabbageA = 1
        - FarmerB = 0
        - CabbageB = 0

    - id: "The farmer brings the goat on the bank B of the river"
      conditions:
        - FarmerA == 1
        - GoatA == 1
      effects:
        - weight: 0.5
          event: "The farmer, the wolf, the goat and the cabbage are on the bank B of
the river"
      assignments:
        - FarmerB = 1
        - GoatB = 1
        - FarmerA = 0
        - GoatA = 0

    - id: "The farmer brings the goat on the bank A of the river"
      conditions:
        - FarmerB == 1
        - GoatB == 1
      effects:
        - weight: 0.1
          event: "The farmer, the wolf, the goat and the cabbage are on the bank B of
the river"
      assignments:
        - FarmerA = 1
        - GoatA = 1
        - FarmerB = 0
        - GoatB = 0
```

```
    - id: "The farmer brings the wolf on the bank B of the river"
      conditions:
        - FarmerA == 1
        - WolfA == 1
        - ((CabbageA < 1) && (GoatA == 1)) || ((CabbageA == 1) && (GoatA < 1))
      effects:
        - weight: 0.5
          event: "The farmer, the wolf, the goat and the cabbage are on the bank B of
the river"
      assignments:
        - FarmerB = 1
        - WolfB = 1
        - FarmerA = 0
        - WolfA = 0

    - id: "The farmer brings the wolf on the bank A of the river"
      conditions:
        - FarmerB == 1
        - WolfB == 1
        - ((CabbageB < 1) && (GoatB == 1)) || (CabbageB == 1) && (GoatB < 1))
      effects:
        - weight: 0.1
          event: "The farmer, the wolf, the goat and the cabbage are on the bank B of
the river"
      assignments:
        - FarmerA = 1
        - WolfA = 1
        - FarmerB = 0
        - WolfB = 0

    - id: "The farmer goes to the bank A of the river"
      conditions:
        - FarmerB == 1
        - WolfB == 1 || CabbageA == 1 || GoatA == 1
        - CabbageB == 1 || WolfA == 1 || GoatA == 1
        - WolfA == 1 || CabbageA == 1 || GoatA == 1
      effects:
        - weight: 0.3
          event: "The farmer, the wolf, the goat and the cabbage are on the bank B of
the river"
      assignments:
        - FarmerA = 1
        - FarmerB = 0

    - id: "The farmer goes to the bank B of the river"
      conditions:
        - FarmerA == 1
        - WolfA == 1 || CabbageB == 1 || GoatB == 1
        - CabbageA == 1 || WolfB == 1 || GoatB == 1
        - WolfB == 1 || CabbageB == 1 || GoatB == 1
      effects:
```

```
        - weight: 0.5
          event: "The farmer, the wolf, the goat and the cabbage are on the bank B of
the river"
      assignments:
        - FarmerB = 1
        - FarmerA = 0
```

All conditions of the events only arise from the problem's nature. Hence the event "The farmer brings the cabbage on the bank B of the river" can happen only if:

- Farmer and cabbage are on the bank A of the river (obvious consideration)
- Wolf and goat are not alone in the same place (wolk would eat the goat)

Likewise, the assignememnts are just the obvius adaptions of the state.

In the proposed problem's definition (there could be others), all events that are not the solution may cause the solution, and all cause-effect relationships have a weight into the interval (0,1). Such approach can be seen as a **brutal force attack** to the problem, a serach of the solution by attempts, since all events can occur multiple times till to cause the solution's event.

Observing the solution found by Joshua makes evident the applied approach.

Table 2. A solution of the Farmer's problem.

Step	Event	Outcome	Is the state changed?	Effects
0	The farmer brings the cabbage on the bank B of the river	No conditions	False	None
1	The farmer brings the goat on the bank B of the river	It happened	True	None
2	The farmer brings the wolf on the bank B of the river	No conditions	False	None
3	The farmer goes to the bank B of the river	No conditions	False	None
4	The farmer brings the cabbage on the bank B of the river	No conditions	False	None
5	The farmer brings the goat on the bank B of the river	No conditions	False	None
6	The farmer goes to the bank A of the river	It happened	True	None
7	The farmer brings the cabbage on the bank B of the river	It happened	True	None

Step	Event	Outcome	Is the state changed?	Effects
8	The farmer brings the goat on the bank B of the river	No conditions	False	None
9	The farmer brings the wolf on the bank B of the river	No conditions	False	None
10	The farmer goes to the bank B of the river	No conditions	False	None
11	The farmer brings the cabbage on the bank B of the river	No conditions	False	None
12	The farmer goes to the bank A of the river	No conditions	False	None
13	The farmer brings the cabbage on the bank A of the river	The event could have happened, but caused a cycle	The event could have changed the state	None
14	The farmer brings the goat on the bank A of the river	It happened	True	None
15	The farmer brings the cabbage on the bank B of the river	No conditions	False	None
16	The farmer brings the goat on the bank B of the river	The event could have happened, but caused a cycle	The event could have changed the state	None
17	The farmer brings the wolf on the bank B of the river	It happened	True	None
18	The farmer goes to the bank B of the river	No conditions	False	None
19	The farmer brings the cabbage on the bank B of the river	No conditions	False	None
20	The farmer brings the goat on the bank B of the river	No conditions	False	None
21	The farmer brings the wolf on the bank B of the river	No conditions	False	None

Step	Event	Outcome	Is the state changed?	Effects
22	The farmer goes to the bank A of the river	It happened	True	None
23	The farmer brings the cabbage on the bank B of the river	No conditions	False	None
24	The farmer brings the goat on the bank B of the river	It happened	True	The farmer, the wolf, the goat and the cabbage are on the bank B of the river

The brutal force attack is recognizable by all failed attempts to make an event happen without having the required conditions.

Steps 13 and 16 highlight how Joshua reasons. Indeed, even if those events could have happend, Joshua discarded them because they would have changed the current state into a configuration already reached in the past. Joshua considered the happening of those events a potential loop.

If you removed all failed attempts from the above table, you have the cleaned solution of the problem.

Table 3. "The farmer, the wolf, the goat and the cabbage"

Step	Cause	effect
1	The farmer brings the goat on the bank B of the river	The state changed
6	The farmer goes to the bank A of the river	The state changed
7	The farmer brings the cabbage on the bank B of the river	The state changed
14	The farmer brings the goat on the bank A of the river	The state changed
17	The farmer brings the wolf on the bank B of the river	The state changed
22	The farmer goes to the bank A of the river	The state changed

Step	Cause	effect
24	The farmer brings the goat on the bank B of the river	The farmer, the wolf, the goat and the cabbage are on the bank B of the river

The proposed approach is interesting, because it highlights the usage of possibilistic cause-effect relationships. Excluding the last step, in all the others the cause-effect relationships partially happened, becase there were the conditions for the cause, but not for the effect. Nevertheless, over the time the partial happening of steps changed the state until the last cause-effect relationship happened completely.

[2] Cfr. https://en.wikipedia.org/wiki/YAML

2. Use Joshua in financial sector

TBD

3. Evolution

TBD

4. User guide

Joshua is available for the following architectures:

- Windows amd64
- Linux amd64
- ARM amd64
- Mac OS/X

The software is available for download at the link:

?????

4.1. How to write a knowledge YAML file

4.2. How to write a problem YAML file

4.3. CLI usage

```
joshua.exe solve -k k_contadino.yml -p p_contadino.yml -m 50 --queuelog
$ ./joshua.exe solve -k ../resources/k_contadino.yml -p ../resources/p_contadino.yml
-m 50 --queuelog
```

5. Other problems

6. The heritage

> A person gets an heritage of 1024€, and it starts spending half of the amount each day. How many days does he take to stay with less than 1 euro?
>
> — anonymous

This simple mathematical problem is interesting because the representing graph is different from that of the problem "The farmer, the wolf, the goat and the cabbage".

Describing it is pretty fast:

- the solution’s event is: "He has less than 1 euro"
- the others events:
  - One day of spending
  - The inheritance is halved

Three cause-effect relationships are enough.

Table 4. Cause-effect relationships

Cause	Effect	Weight
One day of spending	The inheritance is halved	1.0
One day of spending	He has less than 1 euro	0.5
The inheritance is halved	He has less than 1 euro	0.5

Note how this problem:

- there is an always true relationship, indeed when he has one day of spending, then the inheritance is always halved
- there is one event that is the cause of multiple effects

Finally, for this problem the state contains only one variable:

- Inheritance = 1024

7. The water containers

> There are three containers A, B, C. A can hold a maximum of 3 litres of water, and it is empty. B can hold a maximum of 5 litres of water, and it is empty. C can hold a maximum of 8 litres of water, and it is full. There are no marks on the containers. You need to have at least one of B or C with 4 litres of water, how can you do that?
>
> — anonymous

This problem is really similar to "The farmer, the wolv, the goat and the cabbage", even if this one is more related to math and the state plays a crucial role. For this reason, it it may be better starting to define the state.

Table 5. Variables of the state

Variable	Initial value	Meaning
A	0	Amount of litres into container A
B	0	Amount of litres into container be
C	8	Amount of litres into container C
MaxA	3	Maximum amount of litres of the container A
MaxB	5	Maximum amount of litres of the container B
MaxC	8	Maximum amount of litres of the container C

The state may have additional variables, but now it is better to define the events:

- [solution’s event] B or C contains 4
- Empty C in A
- Empty C in B
- Empty B in A
- Empty B in C
- Empty A in B
- Empty A in C

Since the state plays a crucial role to solve this problem, the significant part is defining the assignments due to the events occurrence.

Below table shows the complete definition of the event "Empty C in A".

Table 6. Definition of the event: "Empty C in A"

Condition/assignment	**Expression**	**Condition**
C > 0	Assignment	DeltaA = MaxA - A
Assignment	SubC = min(C, DeltaA)	Assignment
C = C - SubC	Assignment	A = A + SubC

Note that the assignments are a little bit complex, because two temporary variables (DeltaA and SubC) are necessary to represent the final situation after the occurrence of the event.

Finally the definition of the possible relationships.

Table 7. Cause-effect relationships

Cause	**Effect**	**Weight**
Empty C in A	B or C contains 4	0.5
Empty C in B	B or C contains 4	0.5
Empty A in B	B or C contains 4	0.5
Empty A in C	B or C contains 4	0.5
Empty B in A	B or C contains 4	0.5
Empty B in C	B or C contains 4	0.5

Solving this requires a trial-and-error approach, as you can notice looking at the relationships, since the type of all of them is possibilistic.

8. Flights and pilots

> Carlo, Bruno and Alberto work in the same team. One is the pilot, one is the copilot, and one is the engineer. The copilot does not have sisters and brothers, and he earns the least of all. Carlo is the husband of the Bruno's sister, and he earns more than the pilot. Can you say the job of everyone?
>
> — anonymous

A logic problem is probably less intuitive to describe, but it is really interesting, since it highlights Joshua's ability to exploit the concatenation of cause-effect relationships.

TBD

Appendice

Il contadino, il lupo, la capra e il cavolo

The knowledge file.

```
---
  events:
    - id: The farmer, the wolf, the goat and the cabbage are on the bank B of the
river
      statements:
        - FarmerB == 1
        - WolfB == 1
        - GoatB == 1
        - CabbageB == 1

    - id: The farmer brings the cabbage on the bank B of the river
      statements:
        - FarmerA == 1
        - CabbageA == 1
        - ((WolfA < 1) && (GoatA == 1)) || ((WolfA == 1) && (GoatA < 1))
      assignments:
        - FarmerB = 1
        - CabbageB = 1
        - FarmerA = 0
        - CabbageA = 0

    - id: The farmer brings the cabbage on the bank A of the river
      statements:
        - FarmerB == 1
        - CabbageB == 1
        - ((WolfB < 1) && (GoatB == 1)) || ((WolfB == 1) && (GoatB < 1))
      assignments:
        - FarmerA = 1
        - CabbageA = 1
        - FarmerB = 0
        - CabbageB = 0

    - id: The farmer brings the goat on the bank B of the river
      statements:
        - FarmerA == 1
        - GoatA == 1
      assignments:
        - FarmerB = 1
        - GoatB = 1
        - FarmerA = 0
        - GoatA = 0

    - id: The farmer brings the goat on the bank A of the river
```

```
    statements:
      - FarmerB == 1
      - GoatB == 1
    assignments:
      - FarmerA = 1
      - GoatA = 1
      - FarmerB = 0
      - GoatB = 0

  - id: The farmer brings the wolf on the bank B of the river
    statements:
      - FarmerA == 1
      - WolfA == 1
      - ((CabbageA < 1) && (GoatA == 1)) || ((CabbageA == 1) && (GoatA < 1))
    assignments:
      - FarmerB = 1
      - WolfB = 1
      - FarmerA = 0
      - WolfA = 0

  - id: The farmer brings the wolf on the bank A of the river
    statements:
      - FarmerB == 1
      - WolfB == 1
      - ((CabbageB < 1) && (GoatB == 1)) || (CabbageB == 1) && (GoatB < 1))
    assignments:
      - FarmerA = 1
      - WolfA = 1
      - FarmerB = 0
      - WolfB = 0

  - id: The farmer goes to the bank A of the river
    statements:
      - FarmerB == 1
      - WolfB == 1 || CabbageA == 1 || GoatA == 1
      - CabbageB == 1 || WolfA == 1 || GoatA == 1
      - WolfA == 1 || CabbageA == 1 || GoatA == 1
    assignments:
      - FarmerA = 1
      - FarmerB = 0

  - id: The farmer goes to the bank B of the river
    statements:
      - FarmerA == 1
      - WolfA == 1 || CabbageB == 1 || GoatB == 1
      - CabbageA == 1 || WolfB == 1 || GoatB == 1
      - WolfB == 1 || CabbageB == 1 || GoatB == 1
    assignments:
      - FarmerB = 1
      - FarmerA = 0
```

```
  links:
    - type: 0.5
      cause:
        - The farmer brings the cabbage on the bank B of the river
      effect:
        - The farmer, the wolf, the goat and the cabbage are on the bank B of the
river

    - type: 0.5
      cause:
        - The farmer brings the goat on the bank B of the river
      effect:
        - The farmer, the wolf, the goat and the cabbage are on the bank B of the
river

    - type: 0.5
      cause:
        - The farmer brings the wolf on the bank B of the river
      effect:
        - The farmer, the wolf, the goat and the cabbage are on the bank B of the
river

    - type: 0.1
      cause:
        - The farmer brings the cabbage on the bank A of the river
      effect:
        - The farmer, the wolf, the goat and the cabbage are on the bank B of the
river

    - type: 0.1
      cause:
        - The farmer brings the goat on the bank A of the river
      effect:
        - The farmer, the wolf, the goat and the cabbage are on the bank B of the
river

    - type: 0.1
      cause:
        - The farmer brings the wolf on the bank A of the river
      effect:
        - The farmer, the wolf, the goat and the cabbage are on the bank B of the
river

    - type: 0.3
      cause:
        - The farmer goes to the bank A of the river
      effect:
        - The farmer, the wolf, the goat and the cabbage are on the bank B of the
river

    - type: 0.5
```

```
      cause:
        - The farmer goes to the bank B of the river
      effect:
        - The farmer, the wolf, the goat and the cabbage are on the bank B of the
river
```

The state file.

```
---
  variables:
    - name: FarmerA
      value: 1.0
    - name: FarmerB
      value: 0.0
    - name: WolfA
      value: 1.0
    - name: WolfB
      value: 0.0
    - name: GoatA
      value: 1.0
    - name: GoatB
      value: 0.0
    - name: CabbageA
      value: 1.0
    - name: CabbageB
      value: 0.0
  success: "The farmer, the wolf, the goat and the cabbage are on the bank B of the
river"
```

The heritage

The knowledge file.

```
---
  events:
    - id: One day of spending
      statements:
        - Inheritance > 1.0
    - id: The inheritance is halved
      assignments:
        - Inheritance = Inheritance / 2
    - id: He has less than 1 euro
      statements:
        - Inheritance < 1.0

  links:
    - type: 1.0
      cause:
        - One day of spending
      effect:
        - The inheritance is halved

    - type: 0.5
      cause:
        - One day of spending
      effect:
        - He has less than 1 euro

    - type: 0.5
      cause:
        - The inheritance is halved
      effect:
        - He has less than 1 euro
```

The state file.

```
---
  variables:
    - name: Inheritance
      value: 1024
  success: "He has less than 1 euro"
```

The water containers

The knowledge file.

```
---
  events:
    - id: Empty C in A
      statements:
        - C > 0
      assignments:
        - DeltaA = MaxA - A
        - SubC = min(C, DeltaA)
        - C = C - SubC
        - A = A + SubC

    - id: Empty C in B
      statements:
        - C > 0
      assignments:
        - DeltaB = MaxB - B
        - SubC = min(C, DeltaB)
        - C = C - SubC
        - B = B + SubC

    - id: Empty A in B
      statements:
        - A > 0
      assignments:
        - DeltaB = MaxB - B
        - SubA = min(A, DeltaB)
        - A = A - SubA
        - B = B + SubA

    - id: Empty A in C
      statements:
        - A > 0
      assignments:
        - DeltaC = MaxC - C
        - SubA = min(A, DeltaC)
        - A = A - SubA
        - C = C + SubA

    - id: Empty B in A
      statements:
        - B > 0
      assignments:
        - DeltaA = MaxA - A
        - SubB = min(B, DeltaA)
        - B = B - SubB
        - A = A + SubB

    - id: Empty B in C
      statements:
        - B > 0
      assignments:
```

```
        - DeltaC = MaxC - C
        - SubB = min(B, DeltaC)
        - B = B - SubB
        - C = C + SubC

    - id: B o C contiene 4
      statements:
        - B == 4 || C == 4

  links:
    - type: 0.5
      cause:
        - Empty C in A
      effect:
        - B or C contains 4

    - type: 0.5
      cause:
        - Empty C in B
      effect:
        - B or C contains 4

    - type: 0.5
      cause:
        - Empty A in B
      effect:
        - B or C contains 4

    - type: 0.5
      cause:
        - Empty A in C
      effect:
        - B or C contains 4

    - type: 0.5
      cause:
        - Empty B in A
      effect:
        - B or C contains 4
    - type: 0.5

      cause:
        - Empty B in C
      effect:
        - B or C contains 4
```

The state file.

```
---
  variables:
    - name: A
      value: 0
    - name: B
      value: 0
    - name: C
      value: 8
    - name: MaxA
      value: 3
    - name: MaxB
      value: 5
    - name: MaxC
      value: 8
    - name: DeltaA
      value: 0
    - name: DeltaB
      value: 0
    - name: DeltaC
      value: 0
    - name: SubA
      value: 0
    - name: SubB
      value: 0
    - name: SubC
      value: 0
  success: "B or C contains 4"
```