Qualitative Bath

David Speck 12372773 Victor Zuanazzi 12325724

I. Introduction

The goal of this project was to implement a QR engine. When given the qualitative rules, the engine outputs the state graph of the system. We used the QR engine to output the state graph of a simple system consisting of an inlet hose, a container, and an outlet drain. We then use the same reasoner to output the state graph of a more complex version of the same system and compare the results with our hand made state graph.

Most of the times humans reason in a qualitative manner about our surroundings. It is not trivial to formalize it in a way it can be expressed in logical expressions and numeric operations which is required for digital computation. This assignment is an opportunity to get familiar with the challenges regarding designing and developing a QR engine so to allow the computer to reason about a particular system in a similar way as a human would.

II. Hand made causal model and state graph

In order to get some intuition behind the inner workings of a qualitative engine, we have created the state graph for the given problem ourselves. Given the simplicity of the system, we can use the Bernoulli model from fluid dynamics to model relations between the entities. The system is made of three entities, a hose, a container, and a drain as shown in the figure 1 of the causal model.

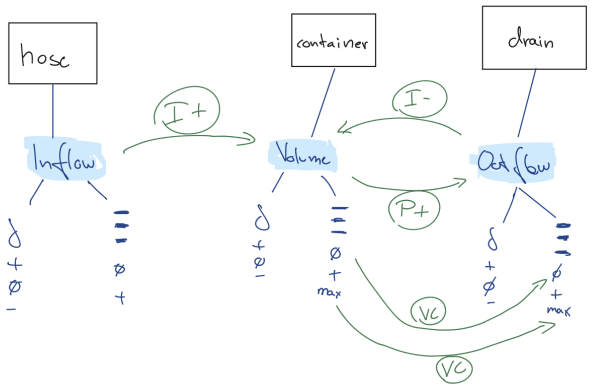

Fig. 1: Causal Model

For the sake of simplicity we have used the following assumptions:

- The hose's inflow is binary. Which means that it is either max either zero.
- When the volume or the outflow reach their max quantities their derivative becomes zero.
- There are no losses in the drain. In other words, the max outflow only happens when the volume is max.
- Only the hose can be operated. In other words, it is not possible for an external influence to change the quantities of the container or of the drain.

By making use of those simplifying assumptions and of the causal model, we derived the state graph shown in figure 2.

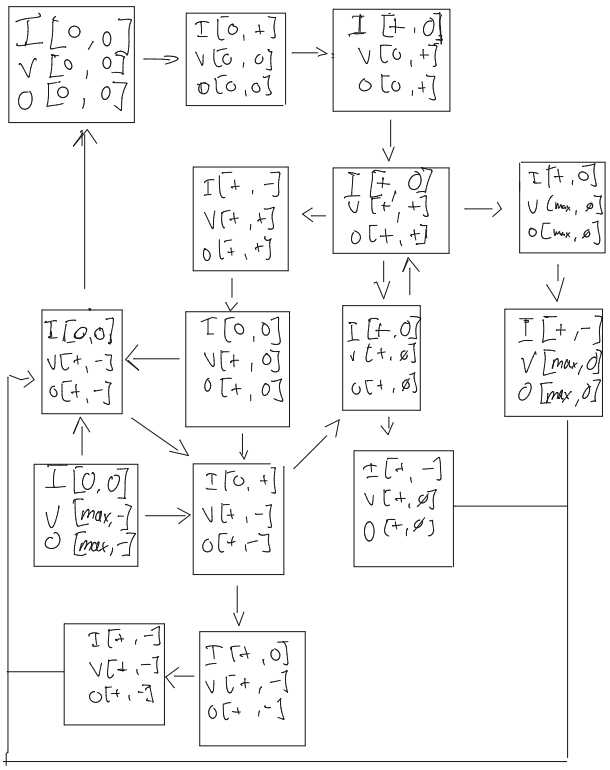

Fig. 2: Hand made state graph

III. QR engine

A. Data structure

We represent states as a dictionary of entities, where each entity is a dictionary of quantities. This double dictionary allows to easily encode the common structural relations in a way that is easy to read and manipulate for the human reader. For quantities, we implemented a custom class which has to member variables: magnitude and derivative, each a custom class on its own. The classes serve the purpose to improve readability of the code and add convenience functionality.

The relevant value of magnitude and derivative are represented as integer enums. This allows for easier manipulation for computation and gives us relations like $<, >$ and $=$ for free. While the enum makes it clear that the values are of

```
{
    'Hose': {
        'Inflow': [+,-]
    },
    'Container': {
        'Volume': [max,0]
    },
    'Drain': {
        'Outflow': [max,0]
    },
}
```

Fig. 3: Example state. A quantity is represented as [magnitude, derivative]

```
{
    "type" : "P+",
    "args" : None,
    "Q1" : ("Container", "Pressure"),
    "Q2" : ("Drain", "Outflow"),
},
{
    "type" : "VC",
    "args" : MValue.MAX,
    "Q1" : ("Container", "Volume"),
    "Q2" : ("Container", "Height"),
},
```

Fig. 4: Two example relations.

qualitative nature and their number representation should be used cautiously.

Further, we choose to represent relations as dictionaries with fixed keys. Each relation has the following keys:

- *type*: a short notation for the type of relation used to map to the functional implementation, e.g. I+, P-, VC
- *args*: additional arguments needed for example for value corresponds relations (VC) to specify the related value
- *Q1*: the key pair that identifies the head quantity of the relation. A tuple of entity name and quantity name used to index a state
- Q2: similar to Q1 a key pair to identify the tail quantity

For simplicity, we modeled bi-direction relations like the value correspondence in the assignment as two directed relations. Additional to influence relations (I+/I-), proportional relations (P+/P-) and value correspondence (VC) we added a relation to notate exogenous influences (EX). These are used to indicate which quantities can receive exogenous influences.

B. Algorithm

Our implementation performs three steps to infer the state graph for a causal model gives as a state blueprint and causal relations. These steps are **state generation**, **state pruning** and **find transitions**. We will go into more detail of each step in the following.

State generation creates all combinatorial states given a blueprint/scheme. For this, we iterate over all possible values for magnitudes and derivatives of all quantities to find all unique combinatorial states.

In the next step, we make use of the causal model to identify valid states and **prune the list of states**. This includes handling upper and lower bound exceptions. For instance, it deletes states in which a quantity has max magnitude and positive derivative.

To apply the causal model we assume the relations come in the order EX, I+/I-, P+/P-, VC. We apply the logic for each relation starting the state to validate and every further relation is applied to all possible resulting states. This is why the order is important since influence relations can change the derivative of quantities which affects the application of proportion relations. Once we applied all relation rules we have set of causal resulting states. If one of the resulting states is the input state, we consider the state as valid.

On the reduced list of states, we **find transitions**. The QR engine searches for two kinds of transitions: exogenous influence and derivative applications.

Exogenous influence are simply the change of a derivative of a quantity that has an EX relation. In our scenario we implemented alternating exogenous signals, meaning whenever the quantity reaches a boundary value and returns to derivative zero we transition to a state that has the derivative changed to +/- and is equals else.

For **derivative transitions** we move the magnitude value into the direction of the current derivative of all quantities. We consider here the difference between interval and point values of the magnitude. If the magnitude has an interval value currently we consider two possible follow up states from then on: the magnitude stays on the interval and the magnitude changes to the next value. The possibilities of all quantities are combined with a Cartesian product.

After applying the derivatives we run the causal model again to model the effect of relations on the transitions. For example, in the bath tube scenario, this ensures that a change in magnitude of inflow impacts the volume derivative immediately. All possible derivative states are validated for continuity and then matched against the list of valid states to find the derivative transitions.

Finally, the found transitions are added to the state dictionaries as a list of state ids of previous and next states with labels for the cause of the transition.

IV. Results

By running the QR engine on the problem specified in the section II, we get the state graph of image 5. It is clear to see the state representation. The state id is shown on top, the quantities as displayed by their first letter, *I* for inflow, *V* for volume and *O* for outflow. Next to the letters, the squared brackets show *[Magnitude, Derivative]*.

As an insightful trace, we included on the edges the changing quantities between two states. For example, the transition between state 12 and state 6 is due to an exogenous influence (exo) and the transition between State 6 and state 9 is due to the derivative (∂) of a quantity changing its magnitude.

We also make explicit states that can loop on itself by adding an edge from the state back to the state. For instance, state 1 represents the state where the container is being emptied, to make explicit that the system can be in this state for an interval of time we added the looping edge. States without the looping edge indicate that the system will immediately transition to another state. As shown in state 2, the container is full and being emptied, which can only last one instant.

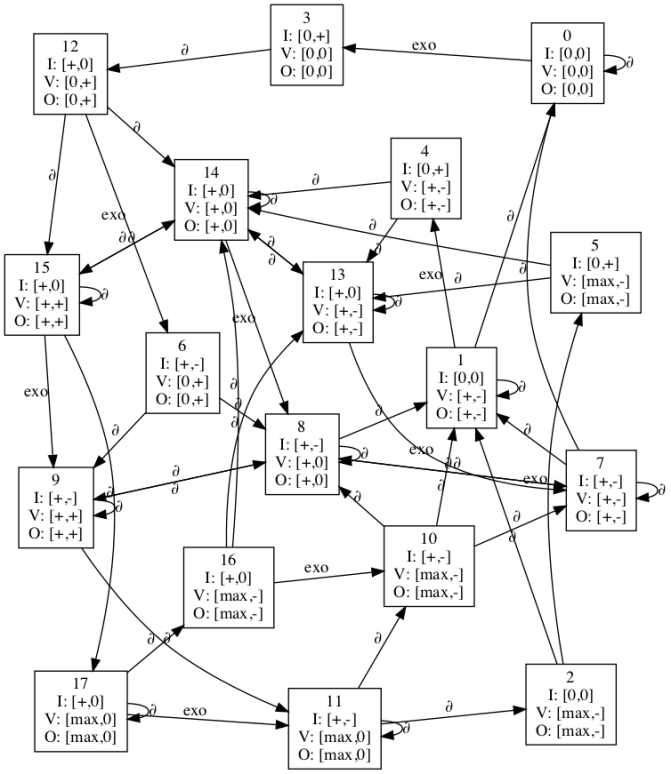

Fig. 5: State graph generated by the QR engine.

We can compare our hand made state graph with the generated one. We did not account for the states 5, 6, 9, 10 and 16. All of them are possible states and, interestingly enough, both state graphs are correct, they were just generated using a slightly different set of assumptions. For instance, state 10 describes the state wherein the exact moment the container became full and someone turned on the tap again. Which we considered impossible in our hand made state graph, but it is plausible under different assumptions. Because of the different number of states, it is not possible to make a one to one comparison between transitions.

We modified the `blueprint` and the `relations` to include height and pressure. Figure 6 shows the resulting state graph. Some implausible states like 15 in the bottom right indicate that the description of the causal model contains errors or our implementation of the QR engine misses still some special cases. Ensuring proper generalization to more complexes model is one way to continue the project.

V. Conclusion

In this project we implemented a generic QR engine, as discussed in the section IV the QR reasoner outputs a valid state graph. Given the qualitative rules, called `relations` and the state `blueprint`, the engine outputs the qualitative state graph of the system. We used the QR engine to output the state graph of two versions of a simple system consisting of an inlet hose, a container, and an outlet drain. We compare the results with our hand made state graph.

By comparing figures 2 and 5 we noticed that we the QR engine has a different set of assumptions then we did when we made our hand made state graph. That is, both of them are possible, but we assumed that no exogenous influence can happen in the same time as an immediate transition, but we did not encode that in out QR engine, which explains most of the differences between results.

The main argument in favor of qualitative reasoners is to create an engine that processes information in a similar way as humans do. Most of the times we reason in a qualitative way about the word, that is relatively hard for a computer, that requires numeric inputs, to do. This assignment gave us a good taste of how challenging it can be to build a generic and robust QR engine.

VI. Attachments

The source code of the project, as well as the results regarding the other metrics, is available at: https://github.com/VictorZuanazzi/Qualitative_Reasoning.git

References

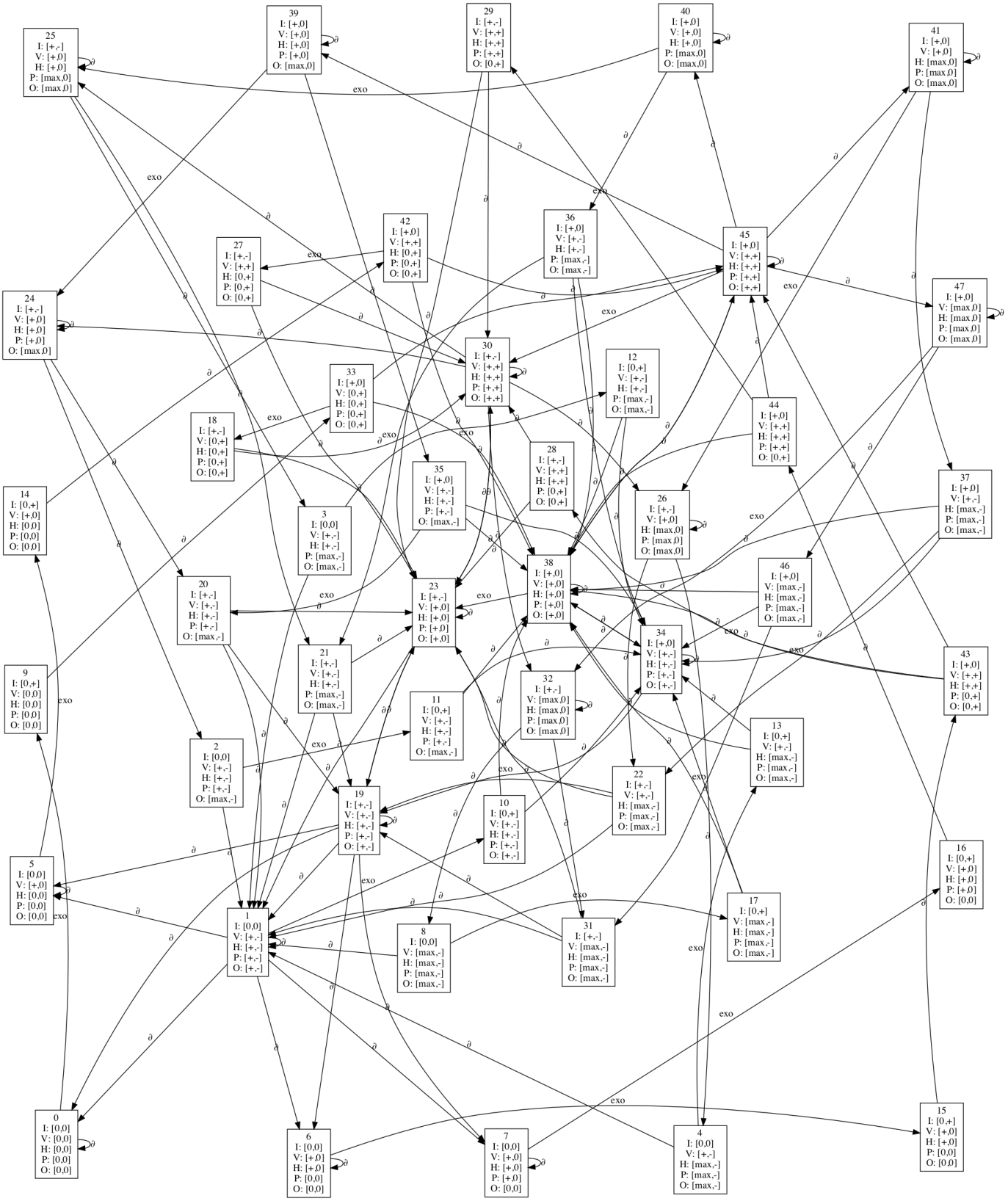

Fig. 6: State graph including Height and Pressure generated by the QR engine.