Understanding phenomena and requirements: the World and the Machine

The World and the Machine
(M. Jackson & P. Zave, 1995)

- Terminology
  - The **machine** = the portion of system to be developed
    typically, software-to-be + hardware
  - The **world** (a.k.a. the environment) = the portion of the real-world affected by the machine
- The purpose of the machine is always in the world

  Examples
  - An Ambulance Dispatching System
  - A Banking Application
  - A Word Processor
  - ...

Example

- For every urgent call reporting an incident, an ambulance should arrive at the incident scene within 14 minutes
- For every urgent call, details about the incident are correctly encoded
- When an ambulance is mobilized, it will reach the incident location in the shortest possible time
- Accurate ambulance locations are known by GPS
- Ambulance crews correctly signal ambulance availability through mobile data terminals on board the ambulances

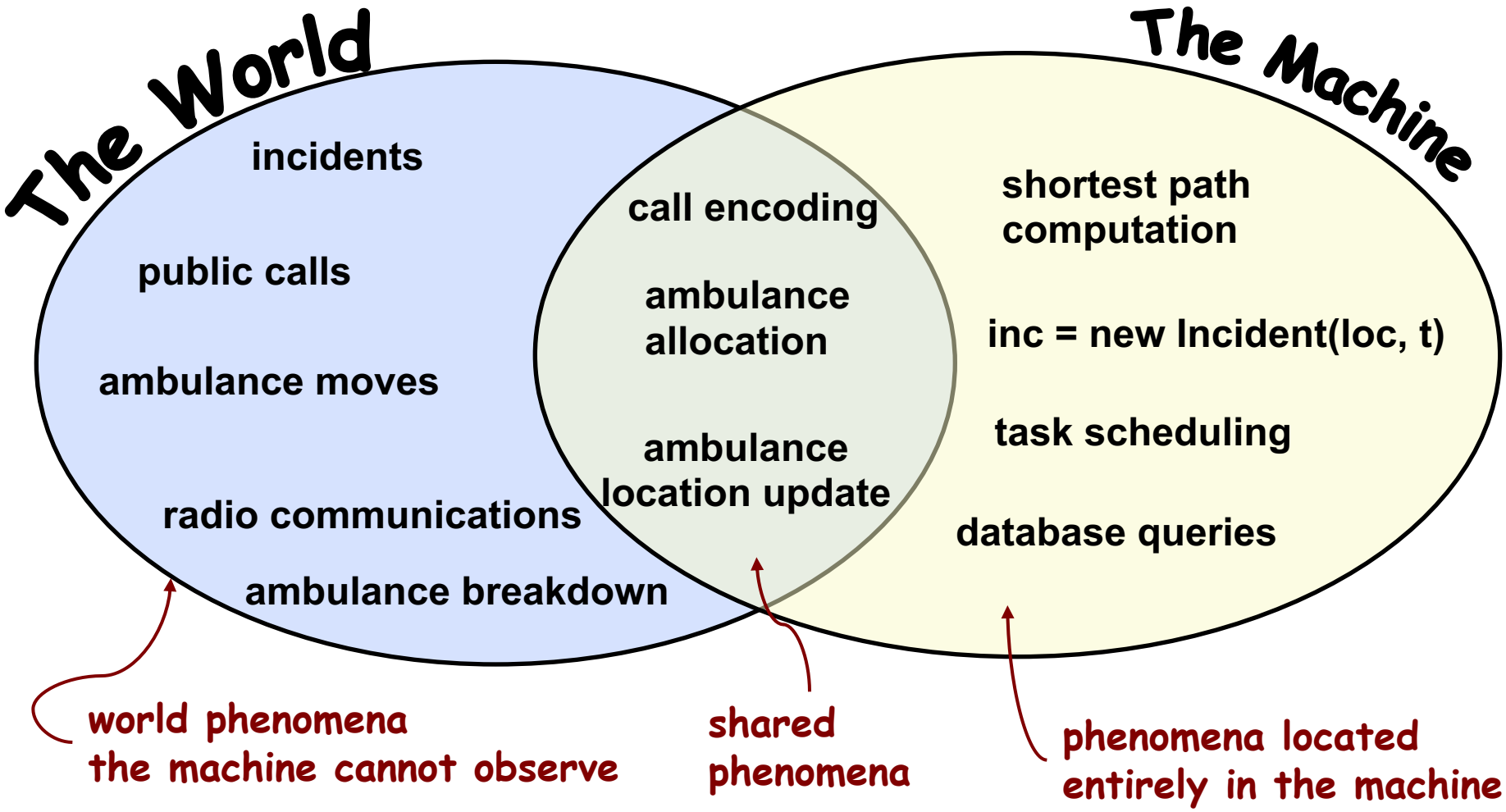
World and machine phenomena

- Requirements engineering is concerned with phenomena **occurring in the world**
  - For an ambulance dispatching system
    - the occurrences of incidents
    - the report of incidents by public calls
    - the encodings of calls details into the dispatching software
    - the allocation of an ambulance
    - the arrival of an ambulance at the incident location
- As opposed to phenomena **occurring inside the machine**
  - For the same ambulance dispatching system
    - the creation of a new object of class **`Incident`**
    - the update of a database entry
- **Requirements models are models of the world!**

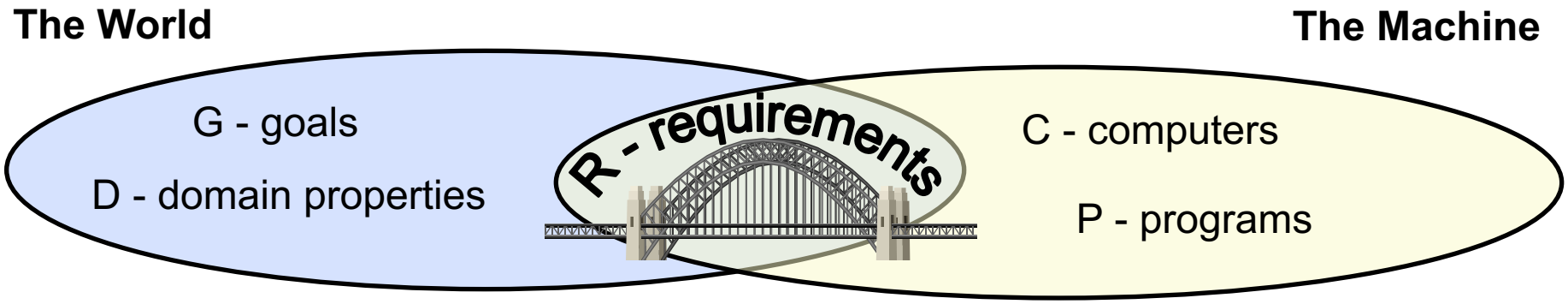
Shared phenomena

- Some world phenomena are **shared** with the machine
- Shared phenomena can be
  - **controlled** by the world and **observed** by the machine
    - e.g., the encodings of calls details into the dispatching software
  - or **controlled** by the machine and **observed** by the world
    - e.g., the allocation of an ambulance to an incident

The ambulance dispatching system



Goals, domain assumptions, and requirements



- **Goals** are **prescriptive assertions** formulated in terms of world phenomena (not necessarily shared)
- **Domain properties**/assumptions are **descriptive assertions assumed to hold** in the world
- **Requirements** are **prescriptive assertions** formulated in terms of **shared** phenomena

Goals, domain assumptions, and requirements

- **Goal:**
  - *'For every urgent call reporting an incident, an ambulance should arrive at the incident scene within 14 minutes'*
- **Domain assumptions:**
  - *For every urgent call, details about the incident are correctly encoded*
  - *When an ambulance is mobilized, it will reach the incident location in the shortest possible time*
  - *Accurate ambulances' locations are known by GPS*
  - *Ambulance crews correctly signal ambulance availability through mobile data terminals on board of ambulances*
- **Requirement:**
  - *When a call reporting a new incident is encoded, the Automated Dispatching Software should mobilize the nearest available ambulance according to information available from the ambulances' GPS and mobile data terminals*

Requirements completeness?

- The requirements R are **complete** if
  1. **R** ensures satisfaction of the goals **G** in the context of the domain properties **D**

     R and D ⊨ G

     Analogy with program correctness: a Program P running on a particular Computer C is correct if it satisfies the Requirements R

     P and C ⊨ R
  2. **G** adequately capture all the stakeholders' needs
  3. **D** represents valid properties/assumptions about the world

What can go wrong when defining G, R, D?

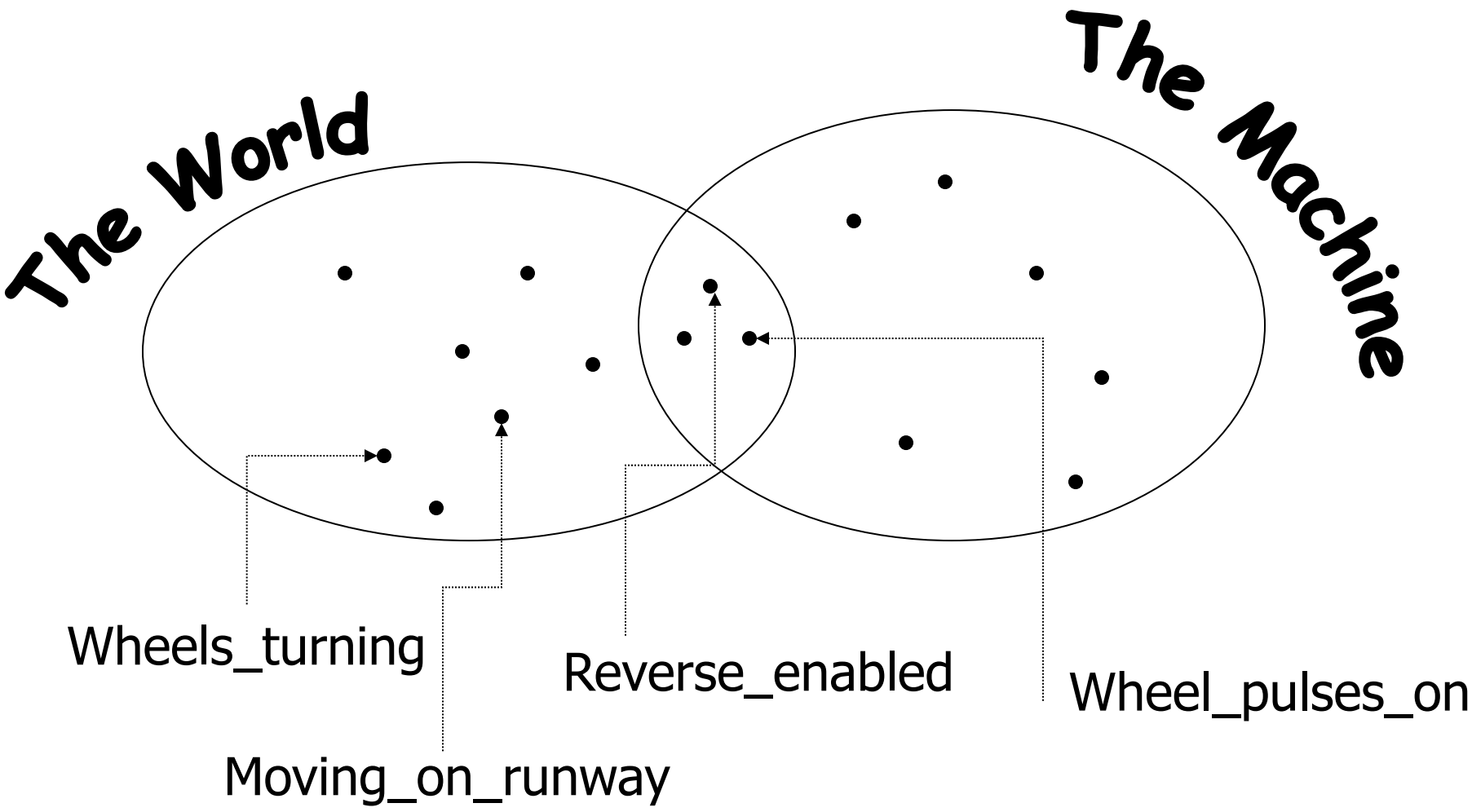
The A320 example

Example – Airbus A320

https://aviation-safety.net/database/record.php?id=19930914-2

- A Lufthansa Airbus on a flight from Frankfurt landed at Warsaw Airport in bad weather (rain and wind)
- On landing, the aircraft's software-controlled braking system did not deploy when activated by the flight crew and it was about 9 seconds before the braking system activated
- There was insufficient runway remaining to stop the plane and the aircraft ran into a grass embankment
- Two people were killed and 54 injured
- Several causes:
  - Human errors
  - Software errors (braking control system)

Example – Airbus A320 Braking Logic



Goal, domain assumptions, and requirement

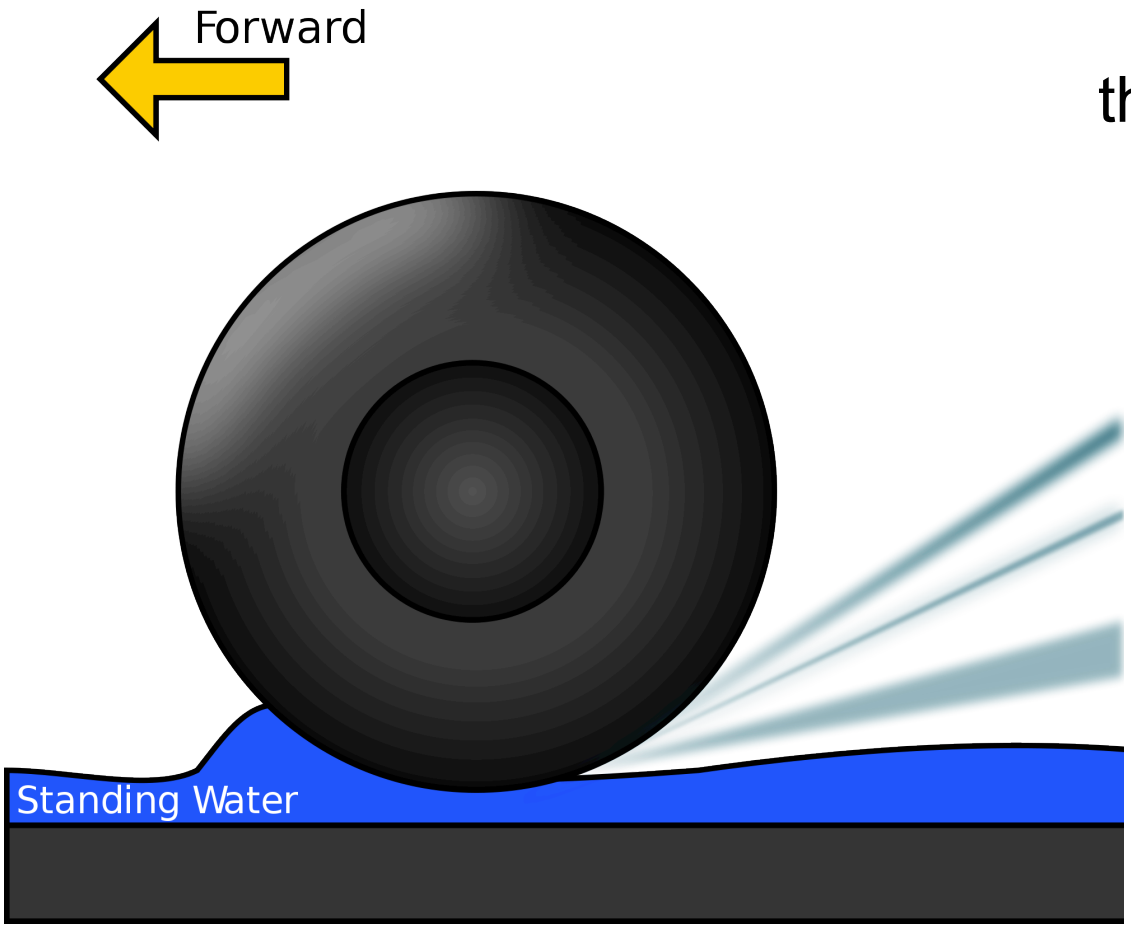
- Goal G:
  - “Reverse thrust shall be enabled if and only if the aircraft is moving on the runway”
- Domain Assumptions D:
  - Wheel pulses on if and only if wheels turning
  - Wheels turning if and only if moving on runway
- Requirements R:
  - Reverse thrust enabled if and only if wheel pulses on
- Verification: R and D $\models$ G applies!

Correctness arguments

- Goal
  - Reverse_enabled $\Leftrightarrow$ Moving_on_runway
- Domain properties
  - Wheel_pulses_on $\Leftrightarrow$ Wheels_turning
  - Wheels_turning $\Leftrightarrow$ Moving_on_runway
- Requirements
  - Reverse_enabled $\Leftrightarrow$ Wheels_pulses_on
- can prove that $R \wedge D \models G$

- … but D are not valid assumptions!!!
- **Invalid domain assumptions -> Warsaw accident**

Ever heard of hydroplaning?



Domain assumptions D do not apply to reality, the correctness argument is bogus!

Incorrect domain assumptions lead to disasters!

Politecnico
di Milano

How to derive requirements from goals: An example

The turnstile example

(M. Jackson, P. Zave, "Deriving Specifications from Requirements: An Example", Proceedings of ICSE 95, 1995)

The turnstile control system

Goals

G1: At any time entries should never exceed accumulated payments (for simplicity, assume 1 coin for 1 entrance)

G2: Those who pay are not prevented from entering (by the "machine")

- They are optative descriptions
- Both are said to be **safety** properties
  - They state that nothing bad will ever occur

Domain analysis (1/2)

- Identify the relevant environment phenomena
  - here: **events**
- "Relevant" with respect to the goals
  - i.e., control people entrance

Domain analysis (2/2)

- **Enter**: puts turnstile back in home position
- **Coin**: coin inserted
- **Push**: frees turnstile
- **Lock**: } electric signals
- **Unlock**: }

environment controlled

machine controlled

shared phenomena

Domain assumptions

- They are real world properties
- They do not depend on the machine

D1: Push and Enter alternate, starting with Push
  - one cannot enter without first pushing
  - one cannot push until the previous visitor entered

D2: Push always leads to Enter
  - enforced by a hydraulics system

D3: If Locked, Push cannot occur

Deriving requirements from goals

- Must find the **constraints on shared phenomena** to be enforced by the machine to achieve the goals
  - G1: At any time, entries should never exceed accumulated payments
  - G1 can be enforced by controlling either entries or coins
    - the machine cannot compel Coin events
    - it can prevent Enter events (through Lock)

How to constrain Enter events?

D1: Push and Enter alternate (starting with Push)

- From D1 we derive (*):

(*) At any time t, if e Enter and p Push events were observed, then $p-1 \leq e \leq p$

- We get (**) by strengthening G1 via (*), by referring to shared phenomena:

(**) At any time t, if p Push and c Coin events were observed, then $p \leq c$

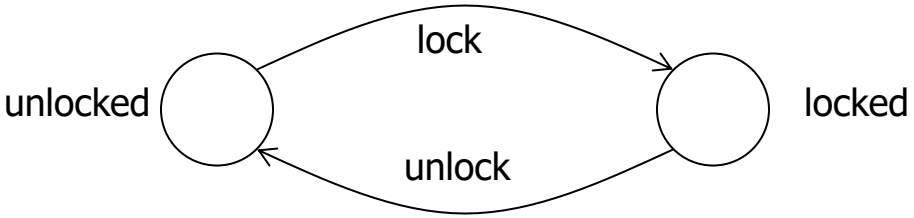
G1: At any time entries should never exceed accumulated payments ($e \leq c$)

if this can be enforced, then G1 holds!

How to ensure p ≤ c?

- When p=c, must prevent further Push until Coin event occurs, … but how can the machine prevent Push?

R1: Impose that Lock and Unlock alternate (initially locked)



R2:

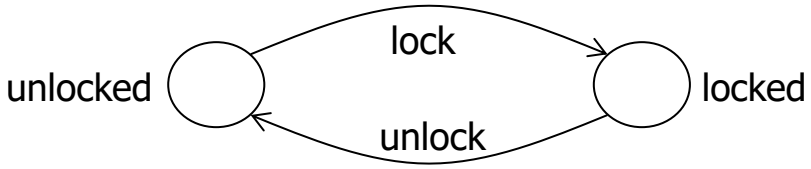
i. If locked and p=c, the machine must not unlock the turnstile, and

ii. If unlocked and p=c, the machine must perform a Lock in time to prevent further Push

Proof that G1 holds

G1: At any time entries should never exceed accumulated payments

- Need to prove that if p=c, no further p can occur if c does not change
- Two possible cases



locked (i.e., l=u+1)

one cannot push (according to R1, R2.i, and D3)

unlocked (i.e. l=u)

immediate lock prevents push (according to R1, R2.ii, and D3)

R1: Impose that Lock and Unlock alternate (initially locked)

D3: If Locked, Push cannot occur

R2.i: If locked and p=c, the machine must not unlock

R2.ii: If unlocked and p=c, the machine must perform a Lock in time to prevent further Push

Deriving requirements from goals

G2: Those who pay are not prevented from entering (by the "machine")

- May be transformed into a further requirement:

R3: i) If unlocked and p<c, the machine does not Lock as long as p<c

ii) If locked and p<c, the machine must perform an Unlock

Proof of G2

G2: Those who pay are not prevented from entering (by the "machine")

R3.i: If unlocked and p<c, the machine does not Lock as long as there is credit

- From R3.i, a new p can occur, hence an e (from D2 and D3)

R3.ii: If locked and p<c, the machine must perform Unlock event

- From R3.ii, we enter case R3.i immediately

D3: If Locked, Push cannot occur

D2: Push always leads to Enter

A warning about terminology

- RE is a relatively young domain, there's no consensus on terminology yet, in particular about what is a requirement
- In Jackson and Zave's work
  - what we call a goal is called a requirement
  - what we call a requirement is called a specification
- Other people (e.g., van Lamsweerde) also use the terms 'System Requirements' & 'Software Requirements'

Observation

- The boundary between the World & the Machine is generally not given at the start of a development project
- The purpose of a RE activity is
  - to identify the real goals of the project
  - to explore alternative ways to satisfy the goals, through
    - alternative pairs (Req, Dom) such that Req and Dom $\models$ G
    - alternative interfaces between the world and the machine
  - to evaluate the strengths and risks of each alternative, in order to select the most appropriate one

Exercise

- Consider the following requirements engineering problem for a pay-per-view system:
- The goal G can be expressed as follows:
  - “Only users who have paid for the service can access application SV (streaming video)”
- We can assume the following valid domain properties D (it is a domain property because it is guaranteed by an existing access control system):
  - “After submitting a payment, personal data and e-mail address, a user receives a password which authorizes access to SV”
- Assume the following requirement R for the “software-to-be”:
  - “Access to SV shall only be granted after a user types an authorized password”

Questions for you

1. Can you formalize such reasoning?
2. What kind of reasoning should you do to prove the correctness of the requirements?
3. What could possibly be wrong in this?

Assignments for the next week class

- Watch the videos you find here
  - On requirement elicitation: https://polimi365-my.sharepoint.com/:v:/g/personal/10143828_polimi_it/EQzGESnSzdBMnTIQvI9XG6QBV0zp715j6HqGqhHJ2Vx7qQ?e=Rx4Cab
  - On modeling: https://polimi365-my.sharepoint.com/:v:/g/personal/10143828_polimi_it/EfuNIeNgPwtIrwcJQNI1s2YBw6G0vEhQDw4yNbpqLH4wQg?e=per3xm
    - Questionnaire available here: https://forms.gle/SjEhrfHC4PQJamyE7
- They will be used as a basis for discussion in class next week