Exercises in Capturing Requirements Constructing Ground Models

- ATM (Cash Machine)
- Vending Machine
- Password Change
- Alarm Clock
- Invoice System
- Lift
- Telephone Exchange
- Double Linked Lists
- Railroad Crossing
- Stack Machine
- Managing Car Rental

Simple ATM (Cash Machine): Problem Description

- Design an ATM, and show it to be well functioning, where the customer is allowed to perform the following ops:
  - Enter the ID.
    - Only one attempt is allowed per session, upon failure the card is withdrawn.
  - Withdraw money from the account.
    - Only one attempt is allowed per session. A warning is issued if the amount required exceeds the balance of the account.
  - Ask for the balance of the account.
    - Allowed only before attempting to withdraw money.
- The central system (supposed to be designed separately) receives the information about every withdrawel and updates the account balance correspondingly.
  - The customer is not allowed to do any other transaction before this update has become effective.

Vending Machine : Problem Description

- Design the control for a ticket vending machine with
  - a coin buffer to temporarily hold money input, a coin tray for money return, a coin store where cashed money is stored
  - a ticket store holding tickets, a ticket tray where to output tickets
  - user operations to
    - choose one among a finite number of possible tickets
    - input the money requested by the system for the chosen ticket
    - cancel the operations done so far & ask for money return
- Refine the machine by allowing
  - a choice of the language for communication with the machine
  - more than one ticket to be chosen in one user operation
  - to insert coins and paper money in portions
  - return of overpaid money

Password Change: Problem Description

- Design a pgm, and show it to be well functioning, which allows a customer to change his password using the following sequence of operations:
  - Enter the user ID and the (current) password.
    - Access should be refused if the ID or the password is incorrect.
  - Enter twice the new password.
    - The new password request should be rejected if the new password is sytnactically incorrect or not correctly repeated. Otherwise the new password should be stored as valid password from now on.
- Refine Password Change by character typing input submachines
- Refine Password Change by restricting attempts to define new pwd

Simple Alarm Clock: Problem Description

- Design an alarm clock, which automatically every second updates & displays currtime, and allows setting of currtime or alarm time (hour, minute) by the user.
  - currtime is displayed at each update
  - to initiate (re)setting curr/alarm time, the user pushes a "time"/"alarm" button
  - the clock, upon reaching the alarm time, rings until
    - either the user cancels the alarm by pressing the "alarm stop" button
    - or 30 seconds have elapsed

    execution of ring may be made dependable upon the position of an alarm-on/off-button, to be set by the user
- Refine the alarm to provide for 3 consecutive ringings with an interval of 5 minutes betweeen them

Incremental Machine Design: Invoice System

J-R Abrial: B-Book, 1996

- Specify a one-user system to handle invoices for orders from clients, and to accordingly maintain the record of products (prices and availability in stock), showing also how to incorporate subsystems for error detection and statistics. Refine the spec by specifying also an appropriate GUI.
  - The system should provide a ("for change") extendable set of operations including the following ones (see details below) :
    - creating/modifying products, clients, invoices
    - reporting errors in handling products, invoices, clients
    - retrieving information on clients, products, invoices and their past
  - For customers' inspection, use abstract & application domain oriented ops, refinable by more detailed equivalent ops
- Illustrate the restrictions of the "one-user one-operation per time system" wrt a asynchronous multiple-user version

Invoice System: Informal Description (cont'd)

- Invoices are issued to clients and have a discount (percentage of the total) and a maximum allowance, both defined upon invoice creation from corresponding attributes of the client. Invoices have a set of positions (e.g. lines), one per ordered product, holding the product's unit price (taken upon creation of the position from the price of the product) & the ordered quantity.
- Clients are classified (e.g. into normal, friend, dubious), yielding a discount (per category), and have a maximum allowance which is applied to each of their invoices.
- Products have price, status (available, sold out), possibly a substitute (another product guaranteed not to be sold out).

Invoice System: Required Properties

- Provide an **error report** when, upon inserting a new product (or its substitute) or upon incrementing its quantity in an invoice:
  - the product is sold out and there is no available substitute
  - the maximum invoice allowance (of the discounted total) would be exceeded by adding the product (or its substitute) to the inv
- Report parameter type errors which may occur in particular when named rules are called
- **Prove** that the system satisfies the following conditions:
  - a sold out product is never made part of an invoice, the system will automatically replace it by its substitute (if there exists one)
  - friend clients (nobody else) get a discount (the same for all), no invoice is ever made for dubious clients
  - (discounted) total of an invoice $\leq$ maximum invoice allowance
  - no invoice lists an ordered product more than once

Lift Control (N.Davis, 1984; see J-R Abrial: B-Book, 1996) : The Problem

- Design the logic to move n lifts bw m floors, and prove it to be well functioning, where
  - Each lift has for each floor one button which, if pressed, causes the lift to visit (i.e. move to and stop at) that floor.
  - Each floor (except ground and top) has two buttons to request an up-lift and a down-lift. They are cancelled when a lift visits the floor and is either travelling in the desired direction, or visits the floor with no requests outstanding. In the latter case,if both floor request buttons are illuminated, only one should be cancelled.
  - A lift without requests should remain in its final destination and await further requests.
  - Each lift has an emergency button which, if pressed, causes a warning to be sent to the site manager. The lift is then deemed ‘out of service’. Each lift has a mechanism to cancel its ‘out of service’ status.

Telephone Exchange: Status of Subscribers

J-R Abrial: B-Book, 1996

- The system controls a network through which pairs among a set of subscribers establish & clear phone conversations
- Subscribers at each moment are in one status out of:
  - free (neither engaged in, nor attempting, any phone convers)
  - unavailable due to a timeout or an unsuccessful attempt to call
  - initiator status :
    - attempting-init attempting to call somebody by dialing his number
    - waiting-init waiting for somebody to answer, after having been connected
    - speaking-init speaking to the called subscriber
  - recipient status :
    - waiting-recip being connected: phone ringing or conversation suspended
    - speaking-recip speaking to the initiator of the phone conversation

Telephone Exchange: System & Subscriber Operations

- Free Subscribers can Lift their handset, thus becoming attempting initiators.
- In every initiator state, initiators (& only them) can Clear their call, thus becoming again free.
- Initiators in attempting status can Dial, trying to complete a subscriber's number.
- The system can MakeUnavailable an initiator, due to a timeout, or due to an unsuccessful attempt to call because of WrongNumber or Failure (called subscriber not free).
- The system can Connect an initiator and a free recipient, making them both waiting (for the recipient to answer).
- A recipient can Answer and Suspend a phone call.

Double Linked Lists : Desired Operations

- Define an ASM which offers the following operations, predicates and functions on double linked lists, whose elements have values in a given set VALUE:
  - CreateList (VALUE) : create a new double linked list with elements taking values in Value
  - Append (L, Val) : append at the end a new element with given value
  - Insert (L, Val, Elem) : insert after Elem in L a new element with Val
  - Delete (L, Elem) : delete Elem from L
  - AccessByValue (L, Val) : return the first element in L with Val
  - AccessByIndex (L, i) : return the i-th element in L
  - empty (L), length (L), occurs (L, Elem), position (L, Elem)
  - Update (L, Elem, Val) : update the the value of Elem in L to Val
  - Cat (L1,L2) : concatenate two given lists in the given order
  - Split (L, Elem, L1, L2) : split L into L1, containing L up to and including Elem, and L2 containing the rest list of L

Double Linked Lists : Desired Properties

- Prove that the Linked List ASM has the following properties:
  - If the next-link of a list element Elem points to Elem', then the previous-link of Elem' points to Elem.
  - L is empty iff the next-link of its head points to its tail.
  - The set ELEM (L) of elements occurring in a list is the set of all E which can be reached, starting from the list head, by applying next-links until the list tail is encountered.
  - After applying Append (L, Val), the list is not empty.
  - A newly created linked list is empty and its length is 0.
  - By Append/Delete the list length in/de-creases by 1.
  - For non empty L and arbitrary elements E the following holds:

    Append (Delete (L,E),E) = Delete (Append (L,E),E)

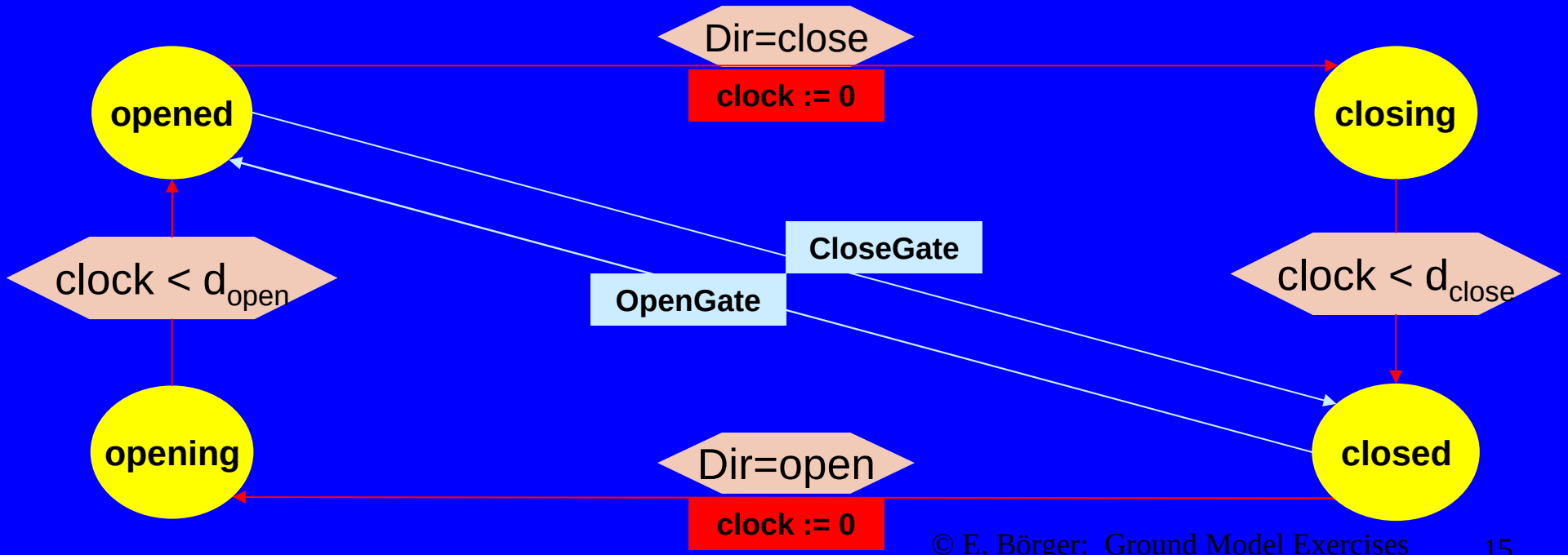
Railroad Crossing Case Study : Problem Description

C. Heitmeyer & D. Mandrioli 1996

- The system to be developed operates a gate at a railroad crossing. A set of trains travel on multiple tracks in both directions. Each track has four sensors, detecting when a train enters resp. exits the crossing: L1, L2 for trains coming from the left, R1, R2 for trains coming from the right. Based on these sensor signals, a controller should signal the gate to open/close.
- Prove the following two properties for the system:
  - Safety: When a train is in the crossing, the gate is closed.
  - **Liveness:The gate is open when no train is in the crossing.**

Railroad Crossing Case Study : Refinement Exercise

- Refine the Railroad Crossing ASM by introducing gate positions between Closed = 0^o and Opened = 90^o. Clarify the real-world timing assumptions which allow that Dir=close/open only when GateStatus=opened/closed.
- Hint. Consider the following timed automaton ASM, with additional GateStatus values closing, opening, and refined rules OpenGate, CloseGate.



Stack Machine : Desired Operations

- Define a Stack Machine as ASM which offers the following instructions, used for executing compiled imperative code:
  - Op: replace the top stack prefix v by Op(v) where Op is a 0-ary or monadic or binary operation and v has length 0, 1, 2 respectively
  - Load(r): push the value which is in register r on top of the stack
  - Store(r): delete the top stack element and copy it into register r
  - Goto(l): transfer control to instruction with label l
  - Cond(l) : transfer control to instruction with label l if Cond is true for the top stack elements (as many as Cond has parameters)
  - Dup: duplicate the top stack element on top of the stack
  - Pop: delete the top element from the stack
  - Halt

  (For a solution see Jbook Fig.9.1)