COMP201 Software Engineering I
Lecture 13 – Formal Specification

Lecturer: T. Carroll

Email: Thomas.Carroll2@Liverpool.ac.uk

Office: G.14

See Vital for all notes

Recap

Recap – Lecture 12

- Introduced the notion of Formal Specification
- Formal Specification Techniques:
  - Algabraic Techniques
  - Model Techniques
- Formal system specification complements informal specification techniques.
- Formal specifications are precise and unambiguous. They remove areas of doubt in a specification.
- Formal specification forces an analysis of the system requirements at an early stage. Correcting errors at this stage is cheaper than modifying a delivered system.
- Formal specification techniques are most applicable in the development of critical systems and standards.
- Example of Algebraic Specification with List ADT

Today

Overview

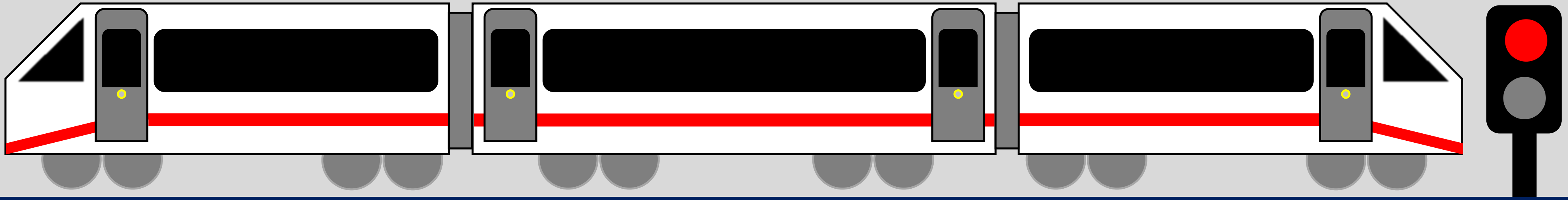
- Recap Algebraic Specification
- Example with 2-Aspect Railway Block Signalling

Algebraic Specification

Example:
2-Aspect Railway Block Signalling

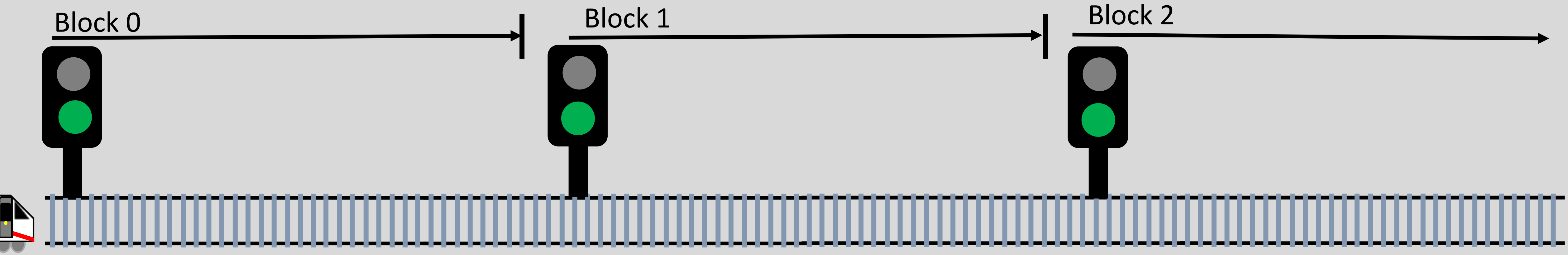
Traffic Lights for Trains….

- 2 aspect signalling uses 2 coloured lights on *signals* to control train movement
  - RED means DANGER (stop)
  - GREEN means CLEAR (go)



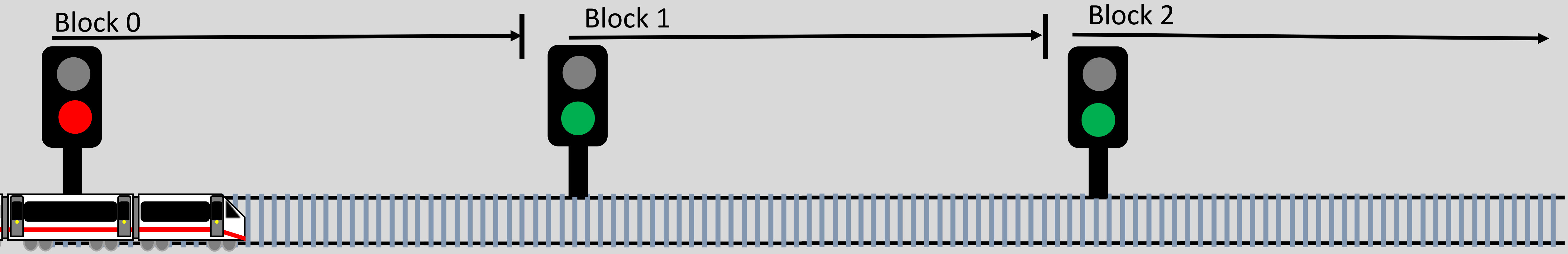
Block by Block…

- Track is split into *blocks,* delimited by signals
- Only one train can occupy a block at any one time
- When a train is detected in a block, then the preceding signal is set to **danger**
- When a block is unoccupied, then the preceding signal is set to **clear**



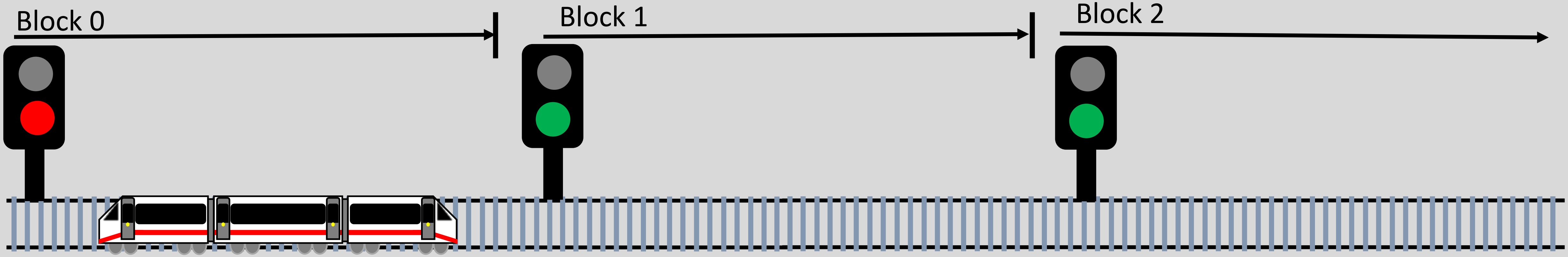
Block by Block…

- Track is split into *blocks,* delimited by signals
- Only one train can occupy a block at any one time
- When a train is detected in a block, then the preceding signal is set to **danger**
- When a block is unoccupied, then the preceding signal is set to **clear**



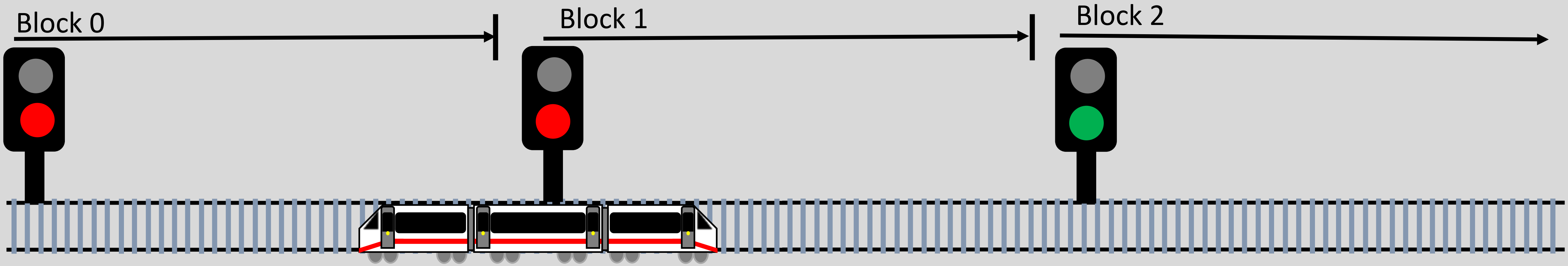
Block by Block…

- Track is split into *blocks,* delimited by signals
- Only one train can occupy a block at any one time
- When a train is detected in a block, then the preceding signal is set to **danger**
- When a block is unoccupied, then the preceding signal is set to **clear**



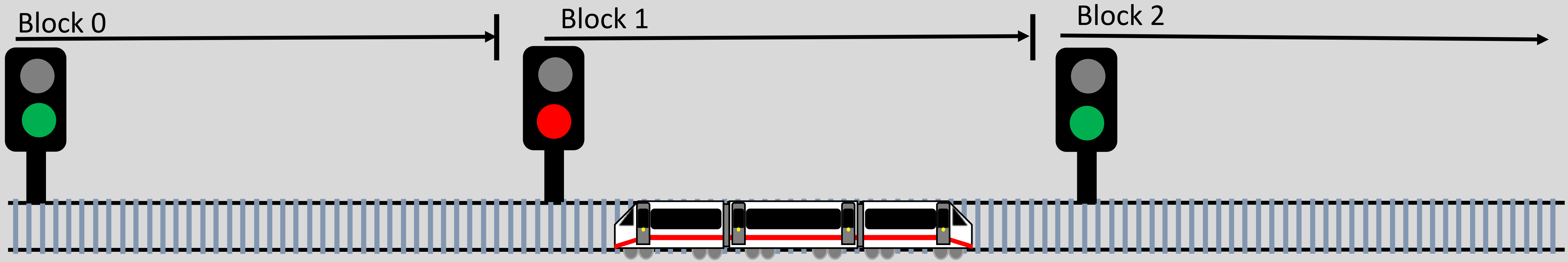
Block by Block…

- Track is split into *blocks,* delimited by signals
- Only one train can occupy a block at any one time
- When a train is detected in a block, then the preceding signal is set to **danger**
- When a block is unoccupied, then the preceding signal is set to **clear**



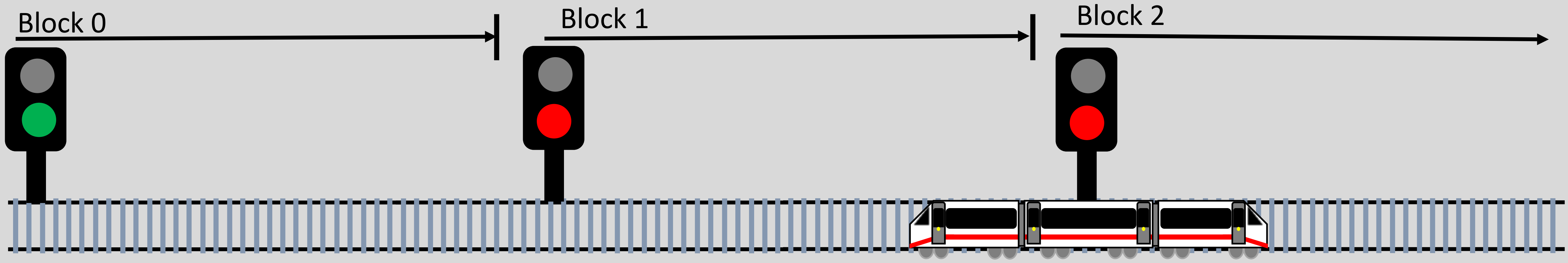
Block by Block…

- Track is split into *blocks,* delimited by signals
- Only one train can occupy a block at any one time
- When a train is detected in a block, then the preceding signal is set to **danger**
- When a block is unoccupied, then the preceding signal is set to **clear**



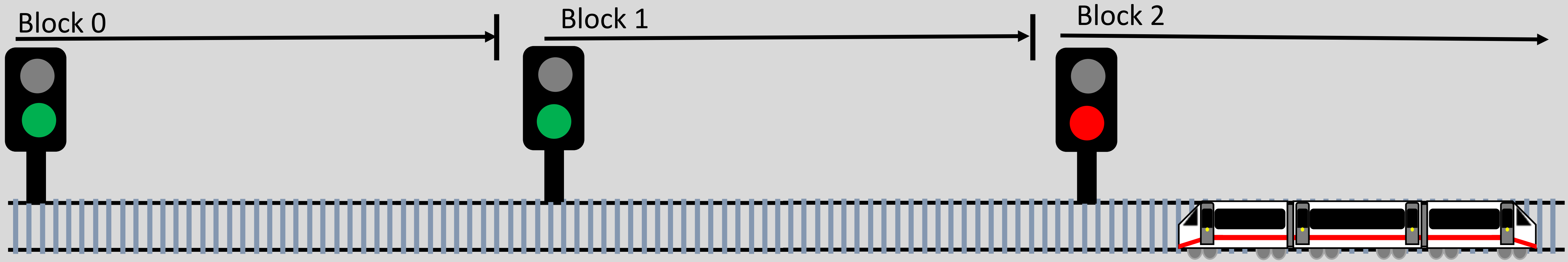
Block by Block…

- Track is split into *blocks,* delimited by signals
- Only one train can occupy a block at any one time
- When a train is detected in a block, then the preceding signal is set to **danger**
- When a block is unoccupied, then the preceding signal is set to **clear**



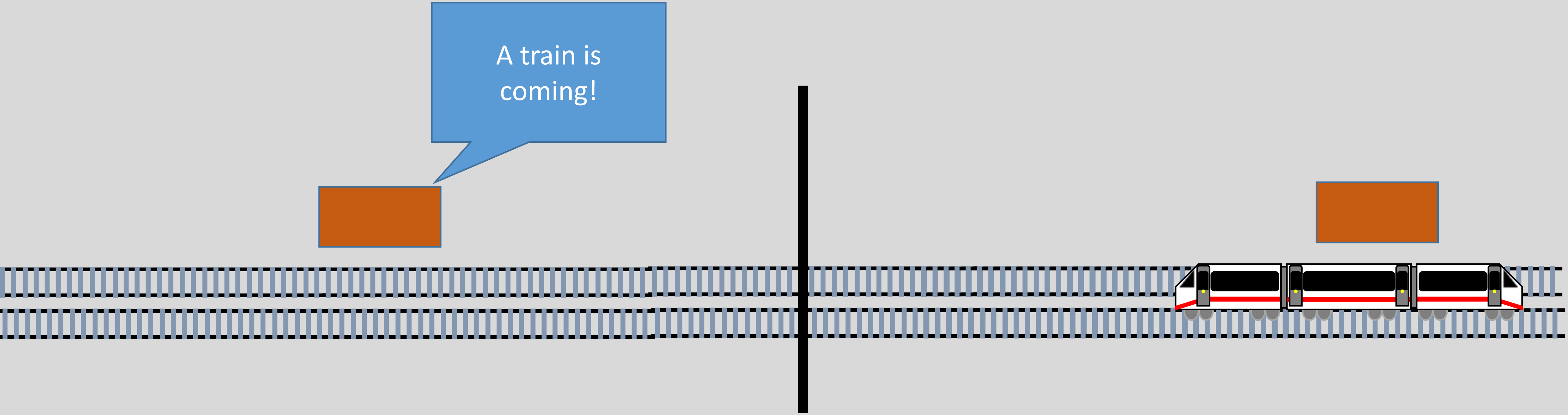
Block by Block…

- Track is split into *blocks,* delimited by signals
- Only one train can occupy a block at any one time
- When a train is detected in a block, then the preceding signal is set to **danger**
- When a block is unoccupied, then the preceding signal is set to **clear**



Signal Boxes Control Many Blocks

- A signal box controls a section of track
- Control of trains is passed from one signal box to the next



Interface Specification in Railway Signalling

- Railway signalling example: Trains move through sectors and blocks of track
- Each sector may include a number of trains but, for safety reasons, **these must be separated** by the **blocks**
- The system should warn the signaller if trains are instructed to move so that the separation rule is breached.

A Sector Object

- Critical operations on an object representing a controlled sector are
  - Enter - Add a train to the controlled sector;
  - Leave - Remove a train from the controlled sector;
  - Move - Move a train from one block to the other;
  - Lookup – Given a train head code (identifier), return its current block;

Primitive Operations

- It is sometimes necessary to introduce additional operations to simplify the specification.
- The other operations can then be defined using these more primitive operations.
- Primitive operations
  - Create - Bring an instance of a sector into existence;
  - Put - Add a train without safety checks;
  - In-space - Determine if a given train is in the sector;
  - Occupied - Given a block, determine if there is a train within it.

Sector Specification (1)

SECTOR

Sort: Sector

Imports: INTEGER, BOOLEAN

Enter – adds a train to the sector if safe to do so

Leave – removes train from the sector

Move – Moves train to another block of safe to do so

Lookup – Finds block location of train in sector

Create – Creates empty sector

Put – Adds train to sector without checks

In-Space – checks if train is already in the sector

Occupied – checks if a specific block is occupied

Enter(Sector, HeadCode, Block) -> Sector

Leave(Sector, HeadCode) – >Sector

Move(Sector, HeadCode, Block) -> Sector

Lookup(Sector, HeadCode) -> Block

Create -> Sector

Put(Sector, HeadCode, Block) -> Sector

In-Space(Sector, HeadCode) -> Boolean

Occupied(Sector, Block) -> Boolean

Sector Specification (2)

```
Enter(S, HC, B) =
        if In-Space(S, HC) then S exception (Train already in block)
        else if Occupied(S, B) then S exception (Block conflict)
        else Put(S, HC, B)

Leave(Create, HC) = Create exception (Train not in sector)
Leave(Put(S, HC1, B1), HC) =
        if HC= HC1 then S else Put(Leave(S, HC), HC1, B1)

…
```

Specification Commentary for Sector

- Use the basic constructors Create and Put to specify other operations.
- Define Occupied and In-space using Create and Put and use them to make checks in other operation definitions.
- All operations that result in changes to the sector must check that the safety criterion holds.

Lecture Key Points

- Formal system specification complements informal specification techniques.
- Formal specifications are precise and unambiguous. They remove areas of doubt in a specification.
- Formal specification forces an analysis of the system requirements at an early stage. Correcting errors at this stage is cheaper than modifying a delivered system.
- Formal specification techniques are most applicable in the development of critical systems and standards.

Lecture Key Points

- Algebraic techniques are suited to interface specification where the interface is defined as a set of object classes.
- Model-based techniques model the system using sets and functions. This simplifies some types of behavioural specification.
- Operations are defined in a model-based spec. by defining pre and post conditions on the system state.