Formal languages and specifications

SEng 321

Validation and verification

Figure 4: Application domain versus machine domain (Adapted from Jackson 1997)

- Validation criteria:
  - Discover and understand all **domain properties**
  - Discover and understand all **requirements**

- Verification criteria:
  - The **program** satisfies the **specification**
  - The **specification**, given the **domain properties**, satisfies the **requirements**

Predicate Logic in Formal RE specifications

- $A \equiv$ a person comes close to the door
- $B \equiv$ the door opens
- $C \equiv$ motion sensor is triggered
- $D \equiv$ OPEN signal to motor

- Reqs: $\{ A \rightarrow B \}$
- Dom: $\{ A \rightarrow C, D \rightarrow B \}$
- Spec: $\{ C \rightarrow D \}$
- **Prove** that Dom $\wedge$ Spec $\rightarrow$ Reqs

Definition and overview of formal methods

A broad view of formal methods includes all **applications of** (primarily) **discrete mathematics** to software engineering problems. This application usually involves **modeling and analysis** where the models and analysis procedures are derived from or derived by an underlying **mathematically-precise foundation**

A formal method in software development is a method that provides **formal language for describing a software artifact** (for instance, specifications, designs, or source code) such that **formal proofs** are possible in principle, **about properties of the artifact** so expressed.

Use of formal methods

- Reasoning about a formal description

- The requirements problem
  – Verification does not eliminate the need for validation!
  – However, the discipline of producing a formal specification can result in fewer specification errors

When to use formal methods

– Mostly functional requirements
– Critical components of the system

Formal Requirements Specification

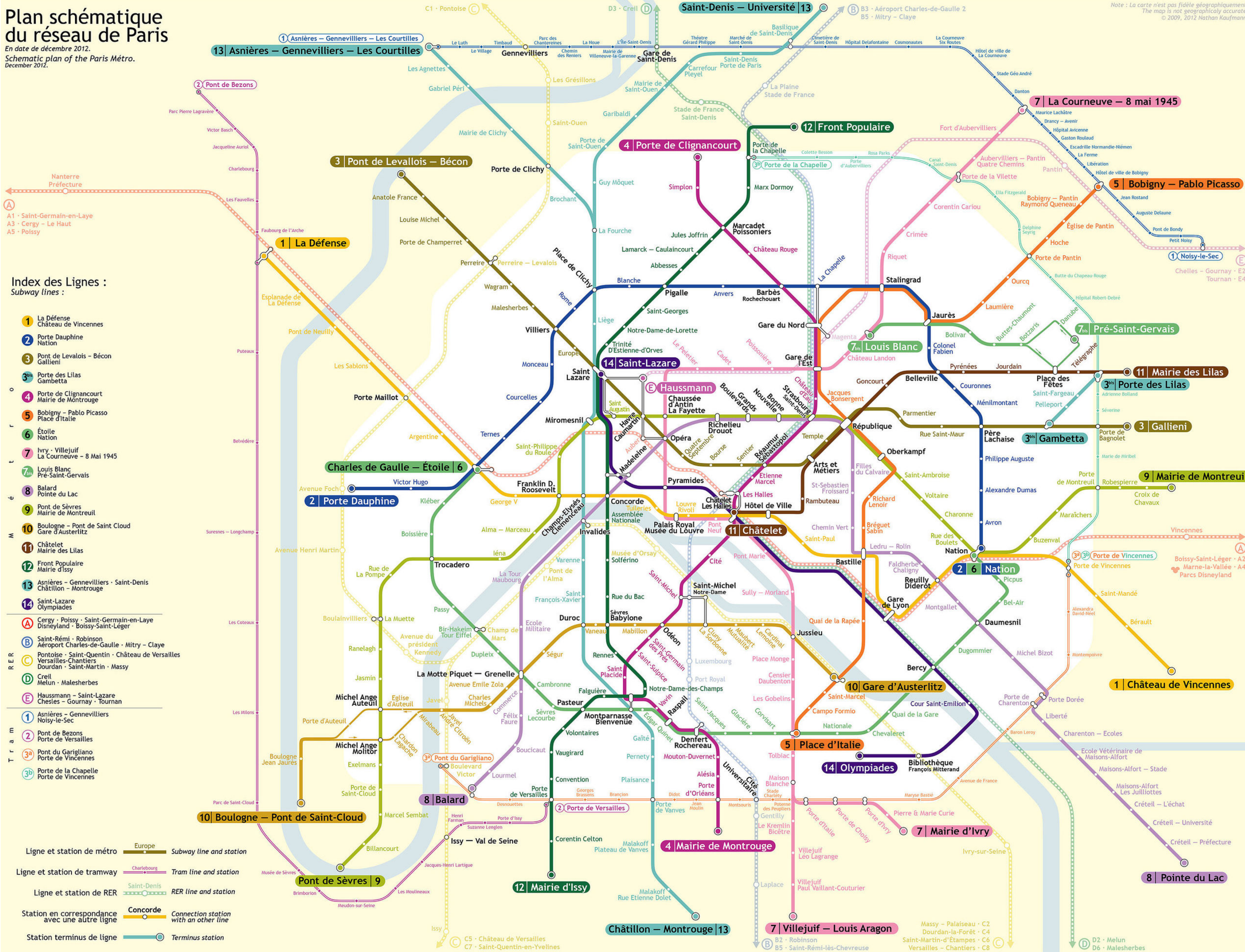
- Formal requirements specifications are specifications that have formal semantics and syntax
- Allows a specification to be precise, unambiguous, and verifiable
- Can be verified automatically

Formal specifications in RE

- Useful in reasoning about the relationships between **domain properties**, **requirement**s and **specifications**
- Prove properties of **requirements** and **specs**
- Good news: very useful when applied properly


Sortie


METROPOLITAIN



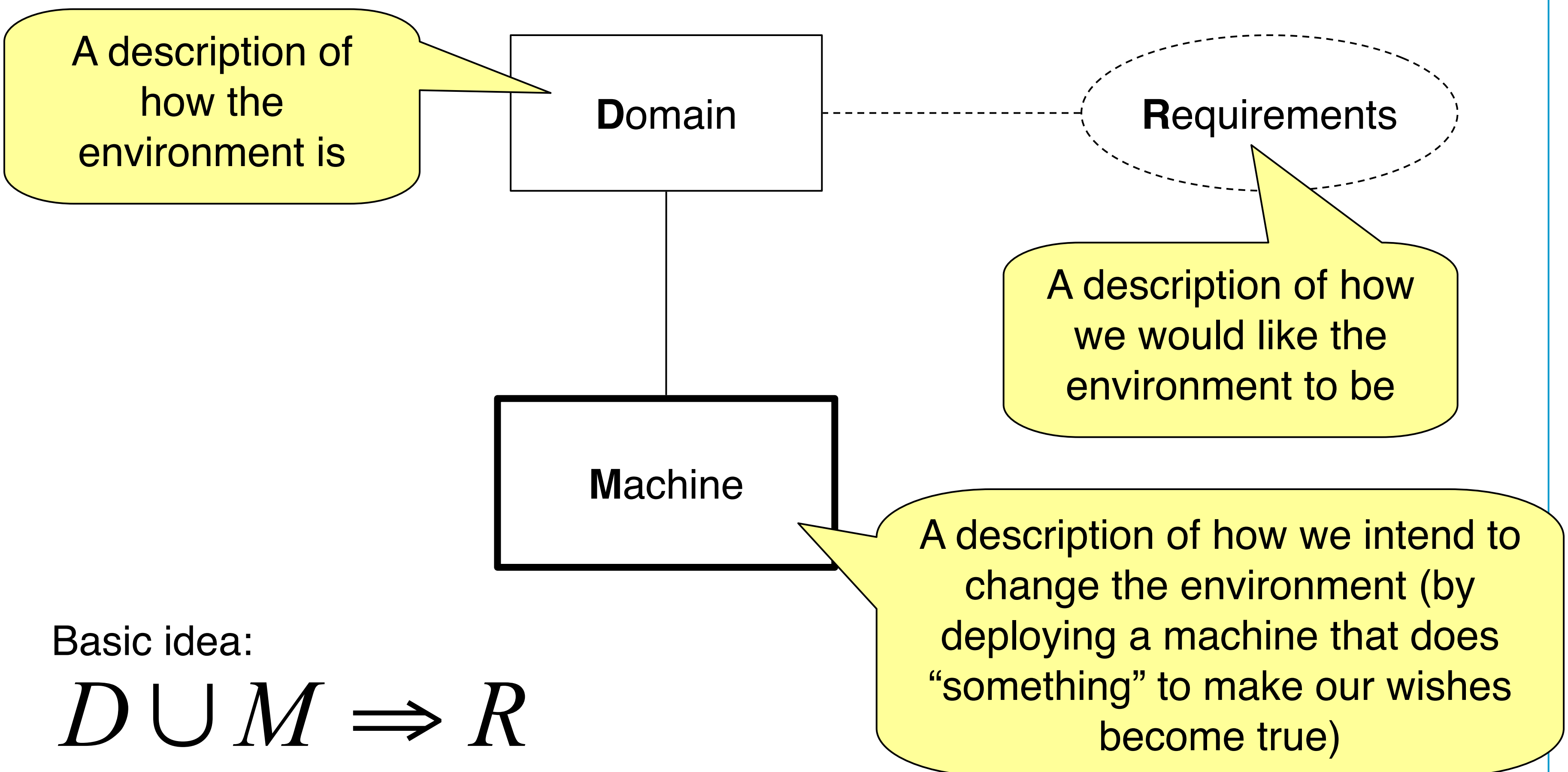
Plan schématique du réseau de Paris
En date de décembre 2012.
Schematic plan of the Paris Métro.
December 2012.
Note : La carte n'est pas fidèle géographiquement
The map is not geographicaly accurate
© 2009, 2012 Nathan Kaufmann
Index des Lignes :
Subway lines :
Métro
1 La Défense · Château de Vincennes
2 Porte Dauphine · Nation
3 Pont de Levalois - Bécon · Gallieni
3bis Porte des Lilas · Gambetta
4 Porte de Clignancourt · Mairie de Montrouge
5 Bobigny - Pablo Picasso · Place d'Italie
6 Étoile · Nation
7 Ivry · Villejuif · La Courneuve - 8 Mai 1945
7bis Louis Blanc · Pré-Saint-Gervais
8 Balard · Pointe du Lac
9 Pont de Sèvres · Mairie de Montreuil
10 Boulogne - Pont de Saint Cloud · Gare d'Austerlitz
11 Châtelet · Mairie des Lilas
12 Front Populaire · Mairie d'Issy
13 Asnières - Gennevilliers - Saint-Denis · Châtillon - Montrouge
14 Saint-Lazare · Olympiades
RER
A Cergy · Poissy · Saint-Germain-en-Laye · Disneyland · Boissy-Saint-Léger
B Saint-Rémi · Robinson · Aéroport Charles-de-Gaulle · Mitry - Claye
C Pontoise · Saint-Quentin · Château de Versailles · Versailles-Chantiers · Dourdan · Saint-Martin · Massy
D Creil · Melun · Malesherbes
E Haussmann - Saint-Lazare · Chesles - Gournay · Tournan
Tram
1 Asnières - Gennevilliers · Noisy-le-Sec
2 Pont de Bezons · Porte de Versailles
3a Pont du Garigliano · Porte de Vincennes
3b Porte de la Chapelle · Porte de Vincennes
Ligne et station de métro · Europe · Subway line and station
Ligne et station de tramway · Charlebourg · Tram line and station
Ligne et station de RER · Saint-Denis · RER line and station
Station en correspondance avec une autre ligne · Concorde · Connection station with an other line
Station terminus de ligne · Terminus station
1 La Défense
1 Château de Vincennes
2 Porte Dauphine
2 6 Nation
3 Pont de Levallois — Bécon
3 Gallieni
3bis Porte des Lilas
3bis Gambetta
4 Porte de Clignancourt
4 Mairie de Montrouge
5 Bobigny — Pablo Picasso
5 Place d'Italie
Charles de Gaulle — Étoile 6
7 La Courneuve — 8 mai 1945
7 Mairie d'Ivry
7 Villejuif — Louis Aragon
7bis Louis Blanc
7bis Pré-Saint-Gervais
8 Balard
8 Pointe du Lac
Pont de Sèvres 9
9 Mairie de Montreuil
10 Boulogne — Pont de Saint-Cloud
10 Gare d'Austerlitz
11 Châtelet
11 Mairie des Lilas
12 Front Populaire
12 Mairie d'Issy
13 Asnières — Gennevilliers — Les Courtilles
Saint-Denis — Université 13
Châtillon — Montrouge 13
14 Saint-Lazare
14 Olympiades
E Haussmann
A1 · Saint-Germain-en-Laye
A3 · Cergy — Le Haut
A5 · Poissy
B3 · Aéroport Charles-de-Gaulle 2
B5 · Mitry · Claye
B2 · Robinson
B5 · Saint-Rémi-lès-Chevreuse
C1 · Pontoise
C5 · Château de Versailles
C7 · Saint-Quentin-en-Yvelines
D3 · Creil
D2 · Melun
D6 · Malesherbes
Massy - Palaiseau · C2
Dourdan-la-Forêt · C4
Saint-Martin-d'Étampes · C6
Versailles · Chantiers · C8
Chelles - Gournay · E2
Tournan · E4
Boissy-Saint-Léger · A2
Marne-la-Vallée · A4
Parcs Disneyland
1 Asnières — Gennevilliers — Les Courtilles
1 Noisy-le-Sec
2 Pont de Bezons
2 Porte de Versailles
3a Pont du Garigliano
3a 3b Porte de Vincennes
3b Porte de la Chapelle

Formal specifications in RE

- Useful in reasoning about the relationships between **domain properties**, **requirements** and **specifications**
- Prove properties of **requirements** and **specs**

- Good news: very useful when applied properly
- Paris metro line 14 entirely controlled by software formally developed by Matra Transport using the **B abstract machine method**

The role of FMs in RE

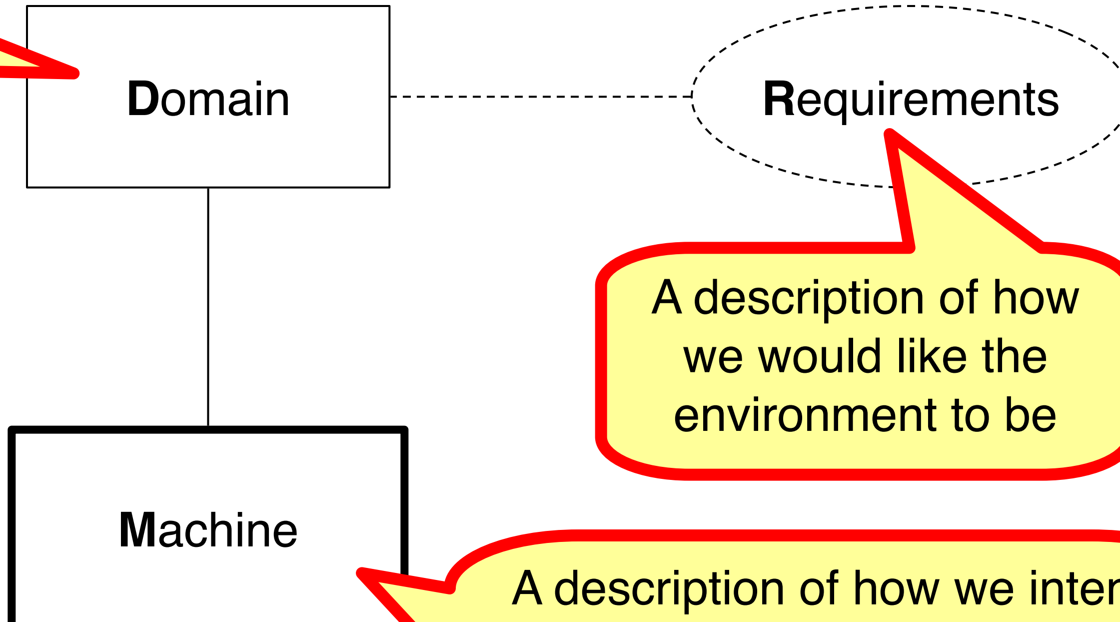


Basic idea:

$$D \cup M \Rightarrow R$$

FMs in RE

All or some of these parts can be **formalized**



A description of how the environment is

A description of how we would like the environment to be

A description of how we intend to change the environment (by deploying a machine that does “something” to make our wishes become true)

Basic idea:

$$\boxed{D \cup M \Rightarrow R}$$

An example: sliding doors

- Design and implement the software for a sliding door controller (SDC) that we want to install in our restaurant (we specialize in stork soup)
- When someone comes close to the door, the door should open automatically
- Etc., e.g.: It should close after that person has entered the restaurant

An example: sliding doors

Requirements
- When a person comes close to the door, the door should open

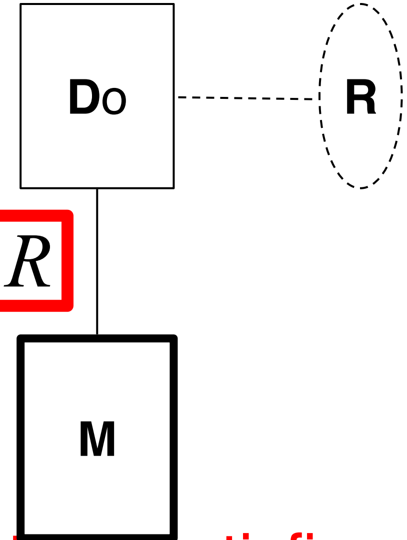
Specification
- When the motion sensor is triggered, the system (SDC) should send the OPEN signal to the motor

Domain
- My restaurant has a motor-controlled door
- A motion sensor is installed above the door
- Motion sensors are triggered by people coming close
- A motor-controlled door opens when the motor receives the OPEN signal

Formalizing SDC in Predicate Logic

- A ≡ a person comes close to the door
- B ≡ the door opens
- C ≡ motion sensor is triggered
- D ≡ OPEN signal to motor

$$\boxed{D \cup M \Rightarrow R}$$



The **specification**, given the **domain properties**, satisfies the **requirements**

- Reqs: { A → B }
- Dom: { A → C, D → B }
- Spec: { C → D }
- **Prove** that Dom ∧ Spec → Reqs

Types of Formal Specifications

- Property-oriented languages
  - System is described by a set of axiomatic properties
- Model-oriented languages
  - System is described by a **state**, and **operations**
  - An **operation** is a function which maps the value of the state and the value of input parameters to a new state

Automated verification

- One major benefit for formal methods is automated verification
- Two major categories of Verification that use Formal Methods:
  - Theorem proving
  - Model checking

Model-oriented languages

- The easiest way to consider a model-oriented language is as a **state machine**
- When the system is in a **certain state**, and is **given a particular input**, it will turn into **another state**
- **Properties** of the system are usually given using a temporal logic
- Want to verify that the **model always holds the properties true**

Model-checking

- An example of a property that can be expressed in temporal logic may be:
  - "if p is true, then it is always the case that q will eventually be true"
- You can verify this behaviour by stepping through a model *M* and checking that the statement above holds

Challenges in applying formal specification methods

- Limited scope (to functional requirements)
- Isolation from other software products and processes in organizations
- Cost (require high expertise in formal systems and mathematical logic)

Example of well-known formal methods and languages

- Language Z
- Communicating sequential processes (CSP)
- Vienna Development Method (VDM)
- Larch
- Formal development methodology (FDM)
- Software Cost Reduction (SCR)