Модели на софтуерни системи

доц. Олга Георгиева

СУ, ФМИ

катедра “Софтуерни технологии”

Лекция 1: Формални методи и подходи

Модел на софтуерна система

Цел и задачи на моделирането

Формални методи

Формална система

Какво е модел?

(*Има ли общоприета дефиниция на модел?)*

All models are wrong – some are useful.

George E. P. Box

Модел

Моделът е *абстракция* или опростено представяне на обекта на изследване във форма, различна от реалното му съществуване.

Моделът отразява или възпроизвежда обекта на изследването в *достатъчна степен*, за да позволи получаване на информация, необходима за неговото изучаване.

Engineering is based on the use of simpler, abstract models for experimentation, reasoning and exhaustive analysis.

Модели – същност и характеристика

Инженерството (в общия смисъл) зависи от използването на **точни** модели за описание на разглежданите системи.

Задачи (ползи) на моделирането:

- **Абстракция,** която позволява да се представят и подчертаят определени детайли**;**

- **Анализ и познание -** начин да се разбере системата, с която работим;

- **Дефиниране и проверка на връзките и взаимодействията** като се постига яснота (недвусмисмислие) в описанията;

- **Проектни решения** – разкриване и разработване на различни решения, свързани с практическата реализация.

Модел на софтуерна система

Модел на софтуерна система е *някаква абстракция* на системата или на част от нея: (част от) спецификация на изискванията, проект, код, тест, поведение на системата.

Моделът е *продукт на нашият разум.* Моделът се възприема във вид на твърдения, формули, математическа символика, схеми и други помощни средства.

Моделите на софтуерни системи - необходимост

Моделите са необходими, за:

- експериментиране, тестване, проверка на проекта преди да бъде реализиран
- Примери:
  - airplane software before test flight
  - net bank services before customer use
  - medical sensor system before patient use
- Моделите могат да представят средата, хардуерни единици, неизвестни части или софтуер на трети лица …

Формални методи (ФМ)

ФМ са математически техники и инструменти за разработване и анализ на софтуерни системи.

В основата е **дискретната математика**, чието развитие през последните няколко десетилетия направи възможно прилагането на математически анализ към компютърните спецификации и приложения.

Формалните методи трябва да се разглеждат като начин за ***подсилване*** **на процеса на разработка** чрез формализация на редица неформални процеси.

Съществуват различни нива на формалните методи по отношение на спецификацията и верификацията.

Ограничения при приложението:

- Математическите методи са добре разработени за последователни системи и за ограничени видове хардуер;
- Оправдано е използването им при *сложни* системи.
- Формалните методи *не могат да решат* цялостно проблема на разработката.

Формалните методи, **основани на елементарната математика, където информацията е структурирана и представена на подходящо ниво на абстракция** се ползват за:

> проектиране,
> разработване,
> тестване,
> експлоатация и поддръжка,
> трансформиране на софтуера.
> документация/спецификация

Примери:

А) **CICS** (*Customer Information Control System*) е фамилия от продукти за банкови трансакции, произведена от IBM UK Laboratories at Hursley Park. Осигурява достъп до данни, комуникация, интегритет, сигурност т.е. за управление на информация...

Б) **B method** – Meteor line 14 на Парижкото метро. 110000 реда на B model, от които се генерира 86000 реда код на Ada без грешка.

Case study: Thierry Lecomte, Thierry Servat, Guilhem Pouzancre, Formal Methods in Safety-Critical Railway Systems (статията е качена на страницата на курса в MOODLE)

- **B Method** was introduced in the late 80s‘
- First real success was Meteor line 14 driverless metro in Paris:

Over 110 000 lines of B models were written, generating 86 000 lines of Ada. No bugs were detected after the proofs, neither at the functional validation, at the integration validation, at on-site test, nor since the metro lines operate (October 1998). The safety-critical software is still in version 1.0 in year 2007, without any bug detected so far.


EMERGENCY DOOR
WAIT

Engineering is based on the use of simpler, abstract models for experimentation, reasoning and exhaustive analysis.

Abstraction is fundamental to Engineering in general, and to Software Engineering in particular.

Формални методи: характеристики

- **Абстракция**

  Abstraction is the most basic principle of software engineering. Abstractions are provided by models.

  ***Model Driven Software Development*, Springer**
  Sami Beydeda, Matthias Book, Volker Gruhn (Eds.)

  Важно!: Съществено качество на добрата спецификация е подходящият избор на ниво абстракция.

- **Доказателството** за характеристики на системата:

  - повишава качеството и осигурява индустриалното приложение на софтуера.

Abstraction? definitions …

- **the act of withdrawing or removing something**
- **the act or process of leaving out of consideration one or more properties of a complex object so as to attend to others**

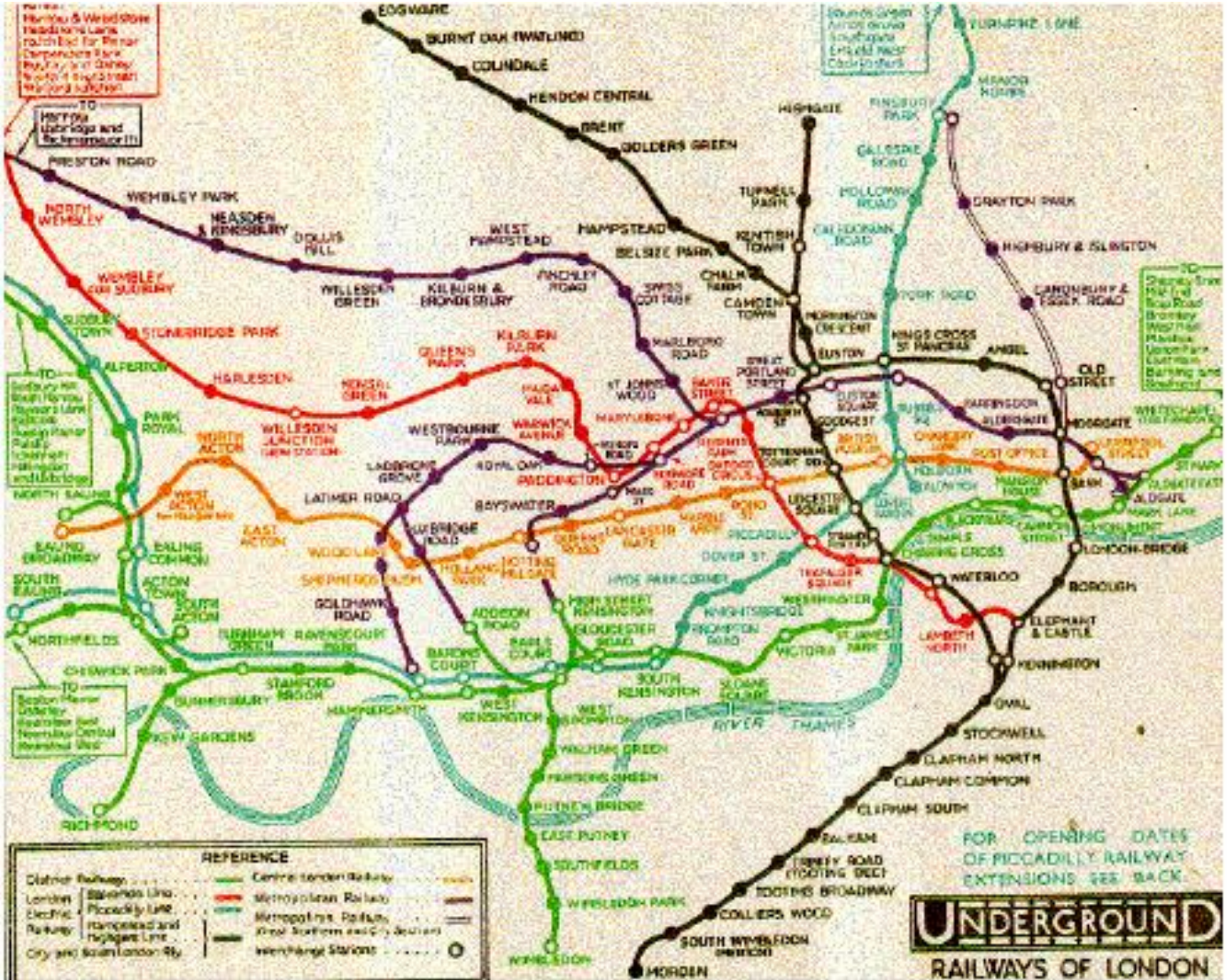
Remove detail (simplify) and focus (selection based on purpose)

- **a general concept formed by extracting common features from specific examples**
- **the process of formulating general concepts by abstracting common properties of instances**

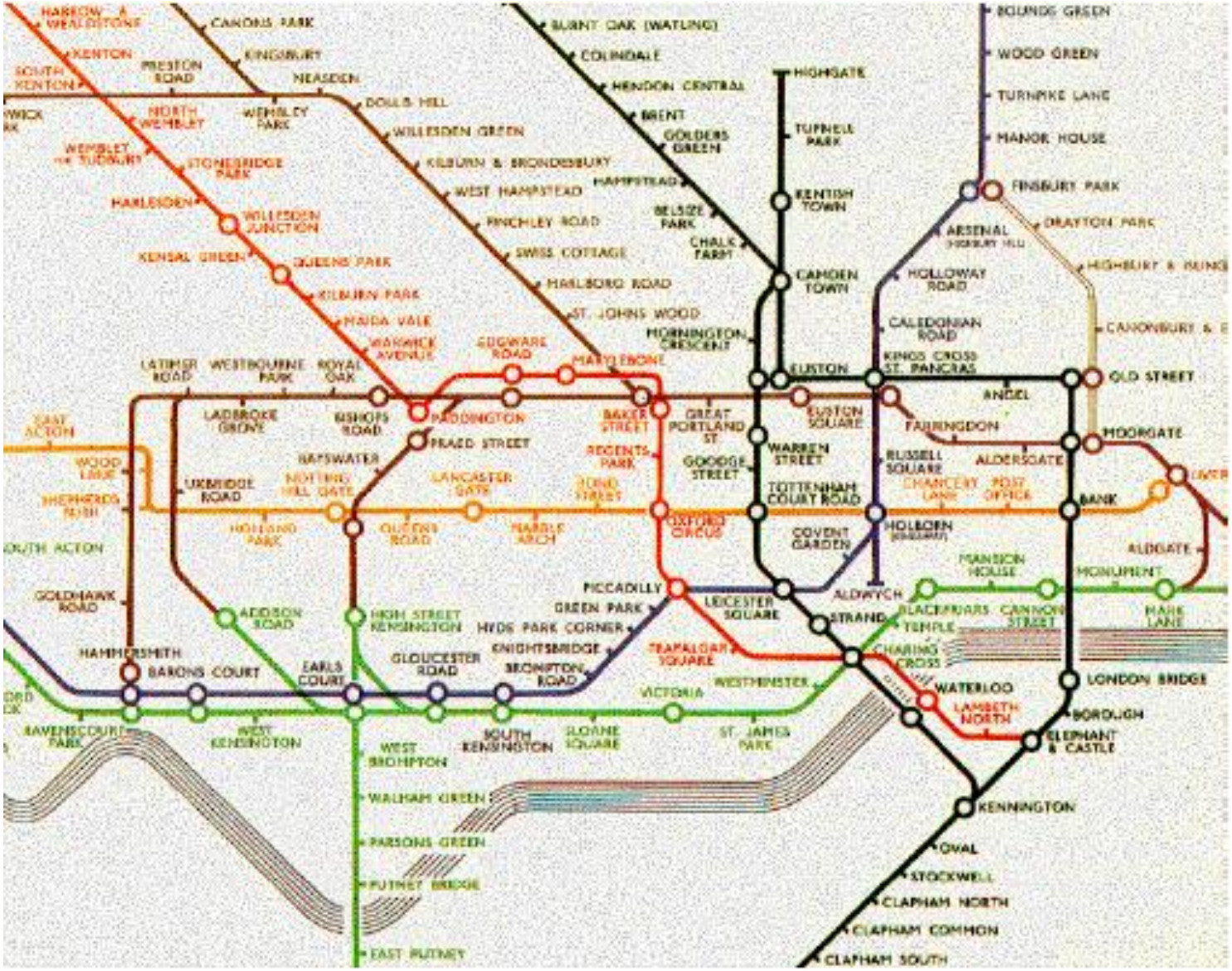
Generalisation (core or essence)

Първата карта (1908) на Лондонското метро била точно копие на географията на трасето.

1930 – London Underground



През 1933 г. картата била променена към по-абстракна илюстрация, наречена Диаграма.



Така Диаграмата е:

- **Абстрактна**
- **Кратка**
- **Пълна**
- **Ясна**
- **С продължителна употреба**
- **Разбираема**
- **Евтина**

Обобщение:

- *Формалното описание ни позволява да проектираме и построяваме системи, защото:*

  Точно описва изискванията

  Открива недостатъците в проектирането

  Реализира рационални приложения

- **Abstraction** allows you to focus on the important aspects of a problem.
  - **Mathematics** allows you to reason about solutions to the problem.

You can earn lots of money if you know the right mathematics!

Формални методи - класификация

- Категоризация в спектъра на формалното описание (**“formality” spectrum)**

- **Полуформални методи (Semi-formal** methods)
  - Use natural language, Diagrams, Tables and simple notation
  - Structured analysis; Object-oriented analysis

- **Formal methods** include:

  - Математически синтаксис и семантика
  - Примери: **Z**, **B**, **VDM**, **LOTOS, CSP** …

Comparison of Specification Techniques

Specification Method	Category	Strengths	Weaknesses
Natural language (Section 11.2)	Informal	Easy to learn Easy to use Easy for client to understand	Imprecise Specifications can be ambiguous, contradictory, and/or incomplete
Entity-relationship modeling (Section 11.5) PSL/PSA (Section 11.4) SADT (Section 11.4) SREM (Section 11.4) Structured systems analysis (Section 11.3)	Semiformal	Can be understood by client More precise than informal methods	Not as precise as formal methods Generally cannot handle timing
Anna (Section 11.9) CSP (Section 11.9) Extended finite state machines (Section 11.6) Gist (Section 11.9) Petri nets (Section 11.7) VDM (Section 11.9) Z (Section 11.8)	Formal	Extremely precise Can reduce specification faults Can reduce development cost and effort Can support correctness proving	Hard for team to learn Hard to use Almost impossible for most clients to understand

Предимства на ФМ за целите на разработването на софтуер

Формалните методи са ниво на абстракция, която позволява да се създаде качествен (удобен, разбираем, надежден) софтуер, защото:

- Намаляват неточността
- Осигуряват строгост в описанията на ранни етапи от разработката
- Могат да проверят коректността и точността, непълнотата и неконсистентността на спецификацията.

Разпространение на ФМ

- ФМ не са широко разпространени

  (FM are slowly emerging from academic research and now being applied in industrial software developments where *safety or security is critical*.)

- **Причини**:

  – Трудности да бъде разбрана спецификацията

  – Трудна формализация на отделни аспекти на системата

  – Трудна възвръщаемост на инвестицията +

  + (консерватизъм и нежелание).

Формални методи

- Осигуряват високо ниво на точност на системата, която да покрие напълно спецификацията

- Не могат да решат всички въпроси и проблеми на разработката и анализа.

Защо !

Основни видове формални методи за спецификация на софтуер

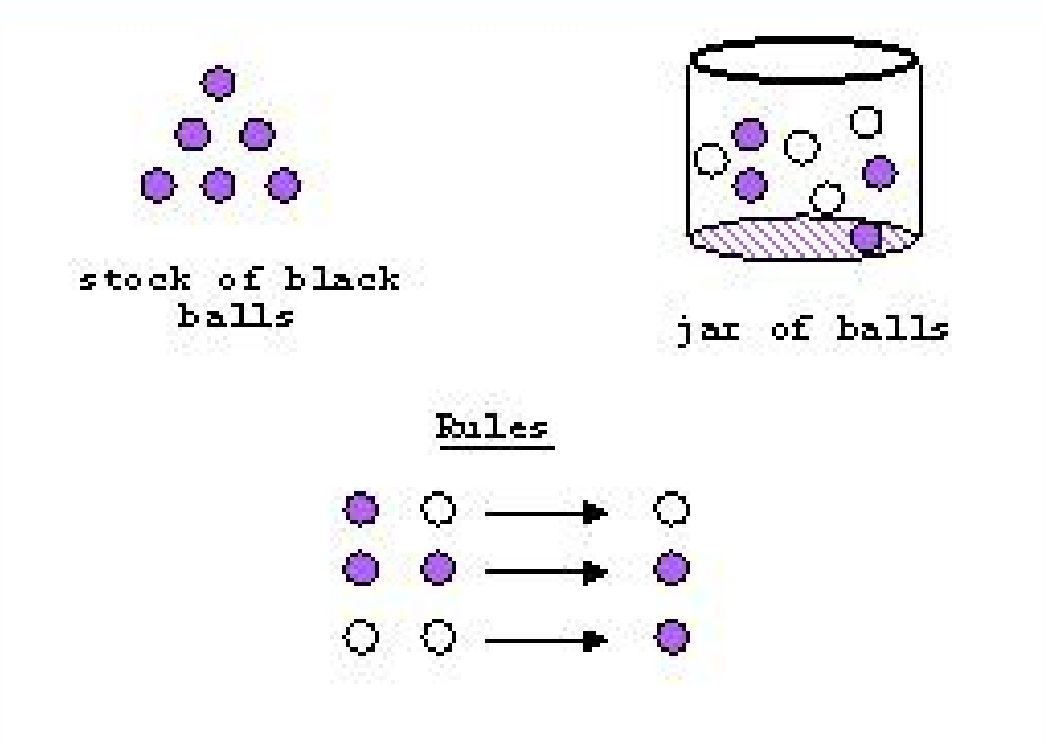
- **Model-based notations**

  Z *(Sprivey, 1989), Vienna Development Method (VDM) (Jones, 1990), B method*

- **Process algebras -based** notations

  Communicating Sequential Processes *(CSP) (Hoare, 1985), CCS (Milner, 1989) and LOTOS (Bjorner, 1987)*

Пример



Задача:

Ако играта е свършила, можем ли да кажем нещо за цвета на последната топка, знаейки оригиналната конфигурация?

Формален модел:

b = черна w = бяла f = преходна функция

$$f(b, w) = \begin{matrix} (b - 2 + 1, w) \\ (b - 1, w - 1 + 1) \\ (b + 1, w - 2) \end{matrix}$$

вадим 2, прибавяме 1 топка

f (0, 1) = (0, 1) f(1, 0) = (1, 0)

Пример

Theorem 1: The game stops

Theorem 2:

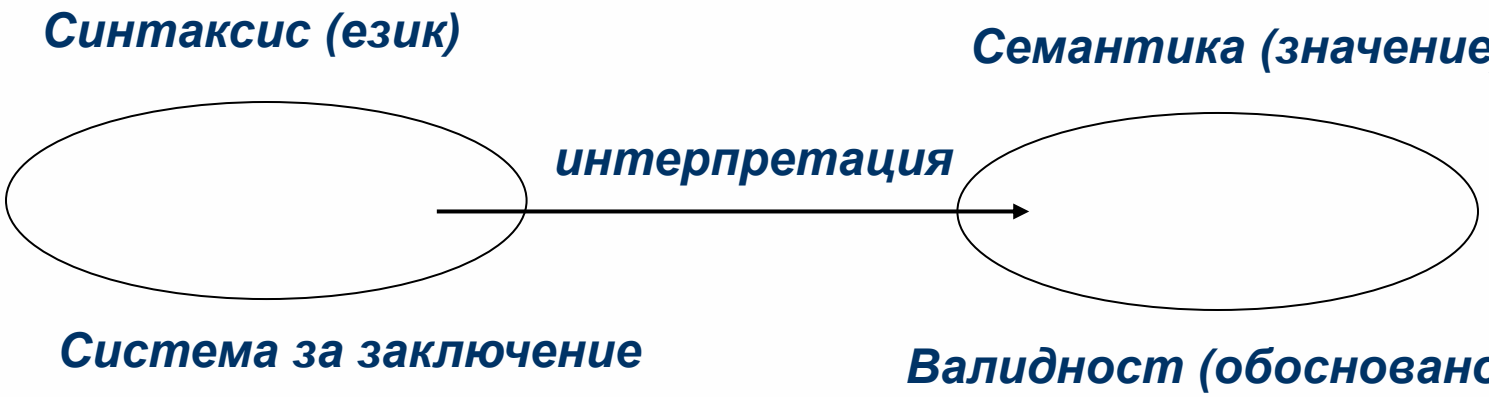
f* (b, w) = if odd (w)
then (0, 1)
else (1, 0)

Компоненти на ФС

- ***Синтаксис (Syntax):*** *специфична нотация за представяне на спецификацията*. Често се базира на синтаксиса на теорията на множествата и предикатните изчисления.

- ***Връзки (Relations):*** дефинират правилата, които индикират, кои обекти правилно/добре удовлетворяват спецификацията.

- ***Семантика (Semantics):*** помага да дефинираме обхвата от обекти, които се използват за описание на системата.



Формална система (ФС)

Неформално основните елементи на формалната система са*:*

1. **Език**, който да може да опише обектите, които ни интересуват

Пр: (b, w), (b - 1, w - 1 +1)

2. **Правила за мислене** (начин на интерпретация (reasoning)) на изказванията в езика:

Пр: (b - 1, w - 1 +1) = (b - 1, w)

3. Начин на определяне на **значението на твърденията/изказванията.**

Пр: b = number of black balls

4. Начин на **представяне и получаване на резултат (разсъждения и интерпретация)**

Пр: facts about numbers:

if b is even then b-2 is even

ФС - дефиниране

Формалната система *се дефинира чрез два основни елемента:*

1) ***формален език (синтаксис)****: азбука + граматика*

2) ***система за извод/заключение****: аксиоми + правила за извод,* ***теореми, доказателство, умозаключение***

Синтаксис: Формални езици

ФС се изграждат чрез формулирането на **Формален език**:

1. **Азбука** – символите, които ще се използват и

2. **Синтактични правила (граматика)** – множество от правила, с които се конструират изречения в дадена система.

Правилно конструираните изречения (твърдения) понякога се наричат ***well-formed formulae* (wff)** на езика.

Пример: *wffs*:

Множествата: (A U B) ∩ (A U C)

Пропозиционна логика: (p => q) ^ (¬ p v q)

Предикатна логика: ∀ x : S ● Stooge(x)

Граматики

Синатктичните правила се нуждаят от свой собствен език за описание, наречен “мета-език (“meta-language”).

Пр. Примерът показва улесняването на изказа в логиката:
"John smokes" can be written as **S(j)**, where **S**=smokes and **j**=John.

• Ще използваме тип като “BNF”

(Backus–Naur Form is one of the earliest metalanguages used in computing and was developed in the 1960s by John Backus and Peter Naur)

Пример:

real = integer | decimal | integer, decimal

decimal = “.” , integer

integer = digit | digit, integer

digit = “0” | “1” | “2” | “3” | ... | “8” | “9”

Примери

Два формални езика с еднакви азбуки, но различни синтактични правила:

1. **Език на реални числа**

   alphabet: {0,1,2,3,4,5,6,7,8,9,.}

   some wffs: “345.678” “2.123” “0.421”

   not wffs: “3.4.5”

2. **Език на числа на разделите (**section numbers)

   alphabet: {0,1,2,3,4,5,6,7,8,9,.}

   some wffs: “3.4.5” “2” “0.421”

   not wffs: “.5” “3.”

(0 Foreword
1 Introduction
2 Methodology
2.1 Counting techniques
2.1.1 Manual procedures
2.1.1.1 Counting apples
2.1.1.2 Counting oranges
…..)

Семантика

• **За да зададем значение на символите**:

> **избираме семантични области** (domain);

> **формулираме интерпретация** (функция на значението), която обяснява как изреченията в синтактичната област придобиват значение (добиват определен смисъл) в семантичната област.

•*Примери*:

Number theory: “2" –> две

Sets: {1,2} сбирка (колекция) съдържаща числата 1 и 2.

Propositional logic: r –> Настоящият месец е октомври.

Predicate logic: *dog*(*Sharo*) –> *Sharo* is a dog.

Системи за извод (Inference Systems)

• **Дедуктивен апарат** за манипулиране на *wffs*.

> Синтактичната манипулация не се отнася до никаква интерпретация

• **Два компонента**:

> **Аксиоми**: *wffs*, които могат да бъдат записани без да се отнасяме (използваме) до някои други *wffs* в езика.

> **Правила за извод/заключение** (Inference rules): правила, които ни позволяват да продуцираме (създаваме) *wffs* като пряко следствие от други *wffs*.

•

Доказателства и теореми

• **Доказателство в една формална система, F, е *последователност* от *wffs* в асоциирания формален език, в която всяко *wff* е**

> **аксиома или**

> пряко следствие от едно или повече предшестващи *wffs,*

определени по правилата на заключение на системата

• Ако *wff* може да бъде доказано/получено т.е. съществува доказателство в **F**, то това wff се нарича **теорема**.

Тогава, ако W може да бъде доказана, пишем : **|- W**

Пример

Нека F е формална система с:

alphabet = {*,@,#}

grammar =

sentence: *stars*, “@”, *stars*, “#”, *stars*

stars = “*” | *stars*, “*”

deductive apparatus =

Axiom: *@*#**

Rule: If a @ b # c is a wff , where a, b, c are *stars*)

then a @ b * # c * is an immediate consequence

Example (continued)

Theorem 1.1: * @ **** # *****

Proof:

1. * @ * # ** Axiom

2. * @ ** # *** Inf rule applied to 1
3. * @ *** # **** Inf rule applied to 2
4. * @ **** # ***** Inf rule applied to 3

Qn: can you prove ** @ *** # ***** ?

Примери:

- Можем да интерпретираме тези символи като:

  * -> 1

  ** -> 2

  ...

  @ -> +

  # -> =

- Тогава какво означава **Теорема 1.1 т.е. * @ **** # ***** ?**

Отг.: 1 + 4 = 5

За тази интерпретация всяка теорема е истина (вярна).

Въпрос?:
Може ли всеки истинен допълнителен факт да бъде доказан в тази система?

Обобщение

Формална система =

= формален език + система за извод

Съгласуваност и пълнота:

- **Интерпретация** на формална система, в която ***wffs*** са изречения, които **могат да бъдат истина или лъжа** е:

> ***съгласувана (консистентна/***sound), ако всяка теорема на системата интерпретира истинно изказване;

> ***пълна*** (complete), ако всяко истинно изказване може да бъде доказано като теорема.

- Повечето формални системи, с които ще се срещаме, са ***съгласувани*** за стандартни интерпретации.

- Много от системите, обаче, не са пълни.

Получаване / извличане:

(*Неформално определение*):

- **Получаване (derivation) на *wff* W** във формалната система **F** от множество **P** от *wffs*, наречени *предпоставки* (premises), е доказателство, в което *wff* в доказателството може да бъде

  > аксиома;
  > *предпоставка*;
  > пряко следствие от предишни *wffs*.

- Получаването на *wff* е **доказателство** във формалната система, чиито аксиоми са били подсилени с предпоставки.

Тогава пишем:

P |- W

т.е. могат да се докажат неверни твърдения.