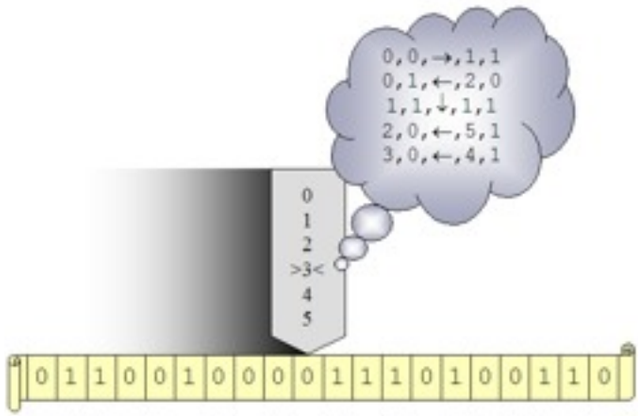
O que é Computação?

Da Máquina de Turing ao Pensamento Computacional



Leila Ribeiro
Instituto de Informática, UFRGS

Sumário

- O que é computação?
- Que problemas podem ser solucionados usando computação?
- História da computação
- A máquina de Turing
- A história continua...
- Que problemas podem ser solucionados usando computação? Como solucioná-los?
- Pensamento Computacional

O que é Computação?

O que é Computação?

- Como resolver problemas?

O que é Computação?

- Como resolver problemas?

- Como fazer com que "máquinas" nos ajudem a resolver problemas?

O que é Computação?

Como resolver problemas?

- Definir abstrações para representar a realidade
- Definir estratégias (procedimentos) para solucionar o problema

Como fazer com que "máquinas" nos ajudem a resolver problemas?

O que é Computação?

Como resolver problemas?

- Definir abstrações para representar a realidade
- Definir estratégias (procedimentos) para solucionar o problema

Como fazer com que "máquinas" nos ajudem a resolver problemas?

- Selecionar máquinas adequadas aos modelos abstratos
- Descrever as estratégias como programas para a máquina selecionada

Solucionando problemas usando computadores...

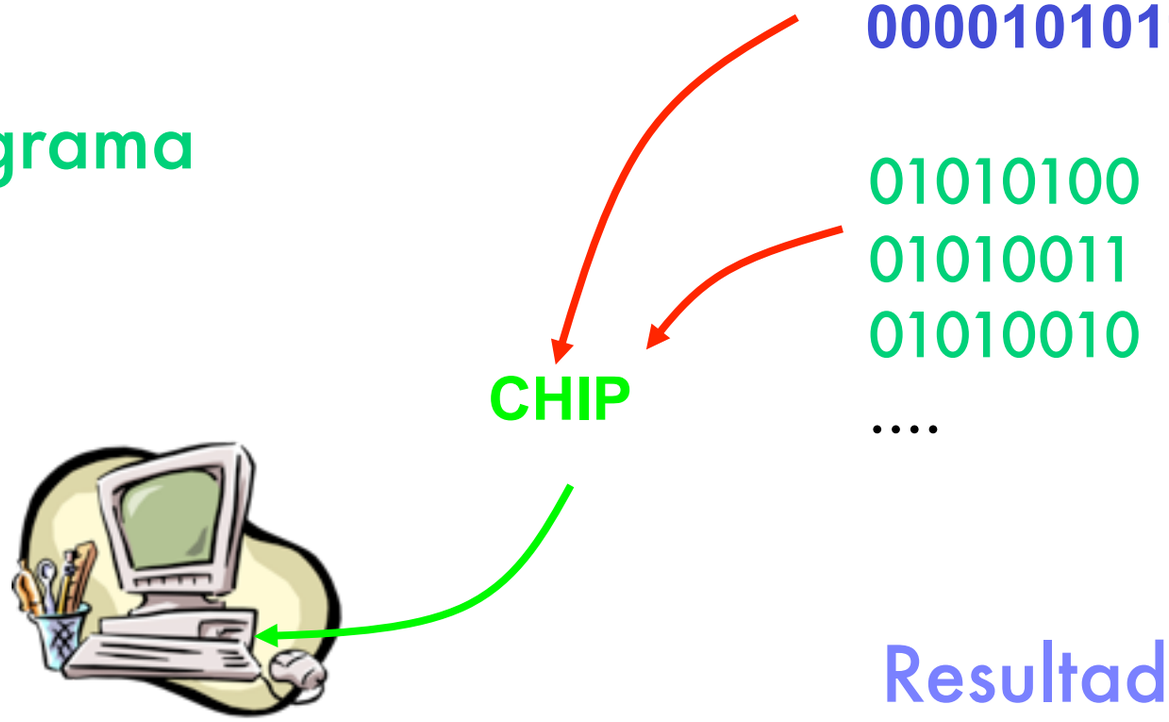
1. Construir **modelo da solução (sistema computacional)**
2. Fazer uma máquina executar este modelo:

Solucionando problemas usando computadores...

1. Construir **modelo da solução (sistema computacional)**
2. Fazer uma máquina executar este modelo:
   - Codificar dados
   - Codificar programa
   - Executar

Solucionando problemas usando computadores...

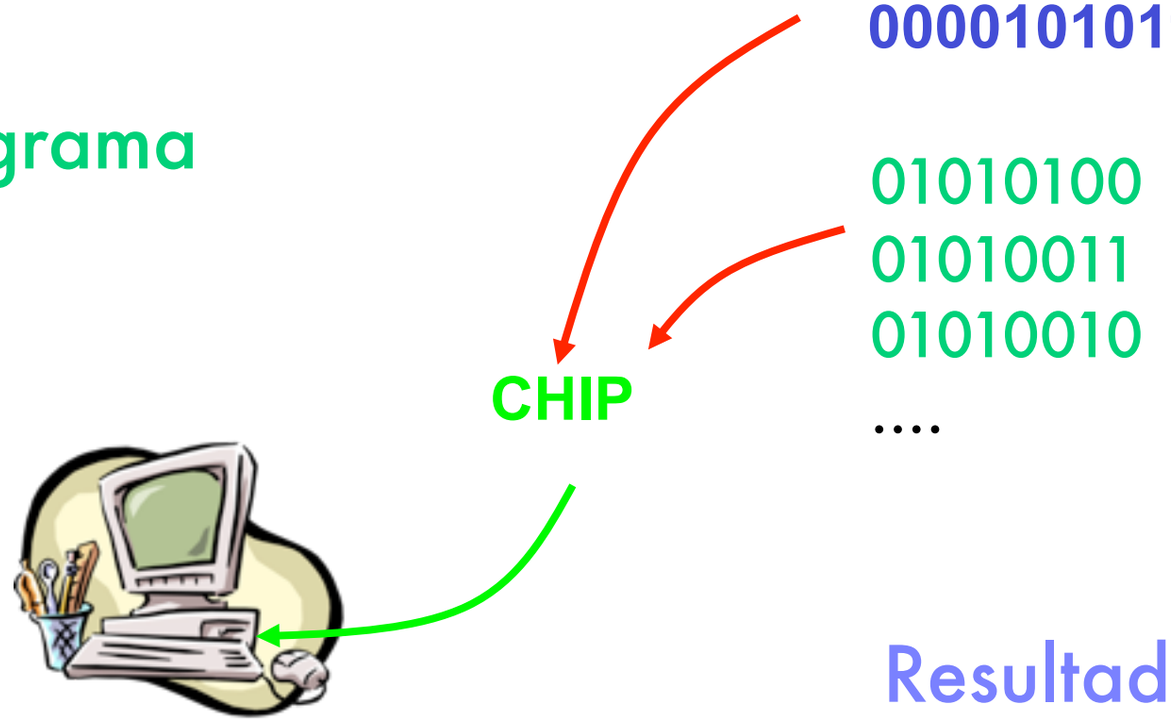
- Codificar dados
- Codificar programa
- Executar



Resultado: 0101000111010O

Solucionando problemas usando computadores...

- Codificar dados
- Codificar programa
- Executar

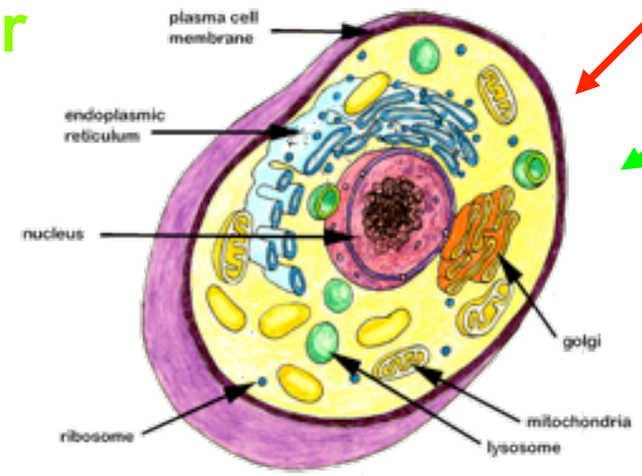


Resultado: 0101000111010O

Mas note que a solução do problema (modelo) é idealmente independente de máquina ou tecnologia...

Solucionando problemas usando DNA...

- Codificar dados
- Codificar programa
- Executar



ATTACGATGC
GCATATATCC

ATTGCCCCCTATTACGTGACGTA
CTTTATAGRCSCCCCCGCGCGCG
CGATGCTGTGTGTGCCCATTATA

Resultado: ATTGCTGTGCCCTA

O que é Computação?

- Como construir as soluções (modelos)???
- Que problemas podem ter suas soluções implementadas em "máquinas inteligentes e obedientes"???

Histórias da Computação

- Hardware : Máquinas

- Software: Algoritmos, que podem ser expressos em diferentes níveis de abstração

Histórias da Computação

- Hardware : Máquinas

  Babbage, Von Neumman, Zuse, Mauchly, Eckert, Wilkes,...
  ENIAC, EDSAC, ACE, UNIVAC, Válvula, Transistor, Circuito Integrado, Cray 1, ...

- Software: Algoritmos, que podem ser expressos em diferentes níveis de abstração

Histórias da Computação

- Hardware : Máquinas

  Babbage, Von Neumman, Zuse, Mauchly, Eckert, Wilkes,…
  ENIAC, EDSAC, ACE, UNIVAC, Válvula, Transistor, Circuito Integrado, Cray 1, …

- Software: Algoritmos, que podem ser expressos em diferentes níveis de abstração

Hilbert, Gödel, Turing, Church, Kleene, Dijkstra, Knuth, Scott, Backus, Floyd, Hopper, Wirth, Plotkin, Hoare, Milner, Pnueli, Petri, Parnas, Jacobson, Booch, Brooks, Clarke, Kiczales, Wing, …

Histórias da Computação

- Hardware : Máquinas

  Babbage, Von Neumman, Zuse, Mauchly, Eckert, Wilkes,...
  ENIAC, EDSAC, ACE, UNIVAC, Válvula, Transistor, Circuito Integrado, Cray 1, ...

- Software: Algoritmos, que podem ser expressos em diferentes níveis de abstração

Hilbert, Gödel, Turing, Church, Kleene, Dijkstra, Knuth, Scott, Backus, Floyd, Hopper, Wirth, Plotkin, Hoare, Milner, Russel, Perlis, Parnas, Jacobson, Booch, Brooks, Clarke, Kiczales, Wing, ...

Métodos para solucionar problemas

Raciocínio Lógico como Base de uma Teoria Matemática

David Hilbert (1900):

Tudo na matemática pode ser provado a partir de axiomas básicos usando raciocínio lógico.

Princípio da Completeza: Todas as perguntas (enunciadas como questões matemáticas) podem ser respondidas (tem prova).

Princípio da Consistência: Existe apenas uma resposta (verdadeiro ou falso) para cada pergunta.

Essa idéia era aceita até 1931...

Teoremas da Indecibilidade

Kurt Gödel (1931):

- Se o conjunto de axiomas que define uma teoria (suficientemente expressiva) é consistente então existem sentenças verdadeiras que não podem ser provadas.

- Não existe procedimento construtivo que prove ser consistente uma teoria axiomática (suficientemente expressiva).

Das Entscheidungsproblem

- *Há uma maneira de saber se existe uma prova para uma sentença?*

Alan Turing (1936)

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

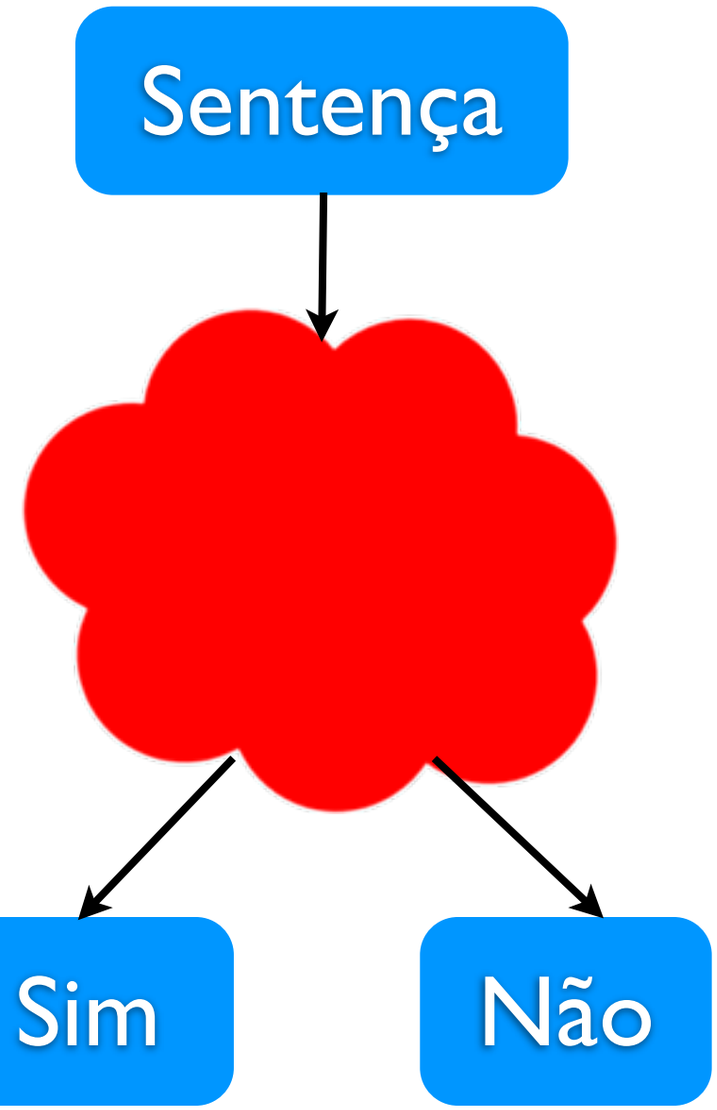
In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers X, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In §8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel [1] . These results {231} have valuable applications. In particular, it is shown (§11) that the Hilbertian Entscheidungsproblem can have no solution.

In a recent paper Alonzo Church[2] has introduced an idea of "effective calculability", which is equivalent to my "computability", but is very differently defined. Church also reaches similar conclusions about the Entscheidungsproblem.[3] The proof of equivalence between "computability" and "effective calculability" is outlined in an appendix to the present paper.

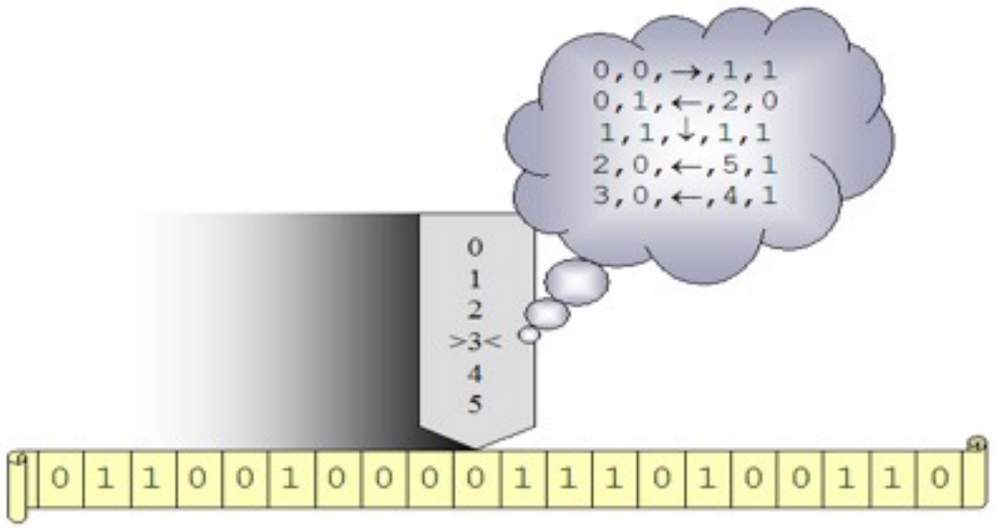
Abordagem de Turing

- *Existe um processo genérico que possa ser usado para determinar se uma sentença tem prova?*
- *Este processo deve ser finito: um número finito de passos que, aplicados a uma entrada finita, entrega depois de um*



Como definir este processo?

A Máquina de Turing



➡ **Modelo matemático** para descrever procedimentos de maneira precisa, mas intuitiva

➡Modelo simples, mas muito expressivo

➡Turing usou os termos **máquina de computar** e **computável**.

➡Se não existe uma máquina de Turing capaz de resolver um problema, ele não é computável!

O Entscheidungsproblem não tem solução!

Outros Modelos

- Cálculo Lambda (Church - 1936)
- Funções recursivas (Kleene - 1936)
- Sistemas de Post (Post - 1943)
- Algoritmos de Markov (Markov, 1954)
- Máquinas de registradores (Shepherdson/Stürgis, 1963)
- DNA computing (Adleman, 1996)
- e muitos outros ...

Outros Modelos

- Cálculo Lambda (Church - 1936)
- Funções recursivas (Kleene - 1936)
- Sistemas de Post (Post - 1943)
- Algoritmos de Markov (Markov, 1954)
- Máquinas de registradores (Shepherdson/Stürgis, 1963)
- DNA computing (Adleman, 1996)
- e muitos outros ...

Todos esses **modelos** são equivalentes:
existem **transformações** de um para outro.

Considerações

- A máquina de Turing é importante porque define a noção de **computabilidade**: define as possibilidades mas também os limites da Computação.

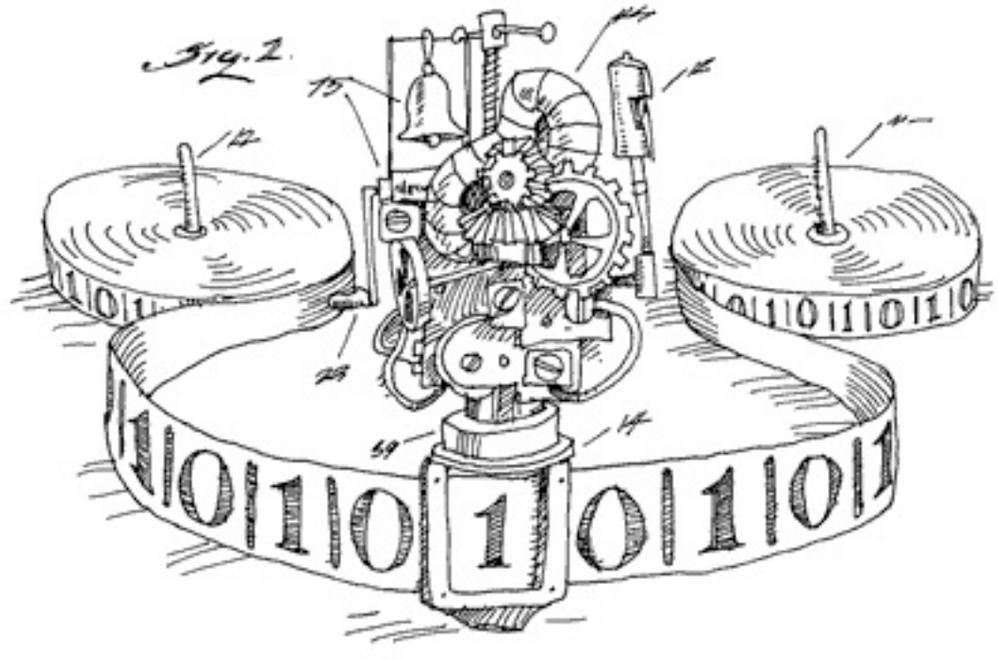
- A máquina de Turing é um modelo teórico, **independente de máquina concreta** (computador, por exemplo).

- Turing também mostrou que existe uma **máquina universal**, que é capaz de simular o comportamento de qualquer outra máquina de computar.

Programas  **Dados**

Considerações

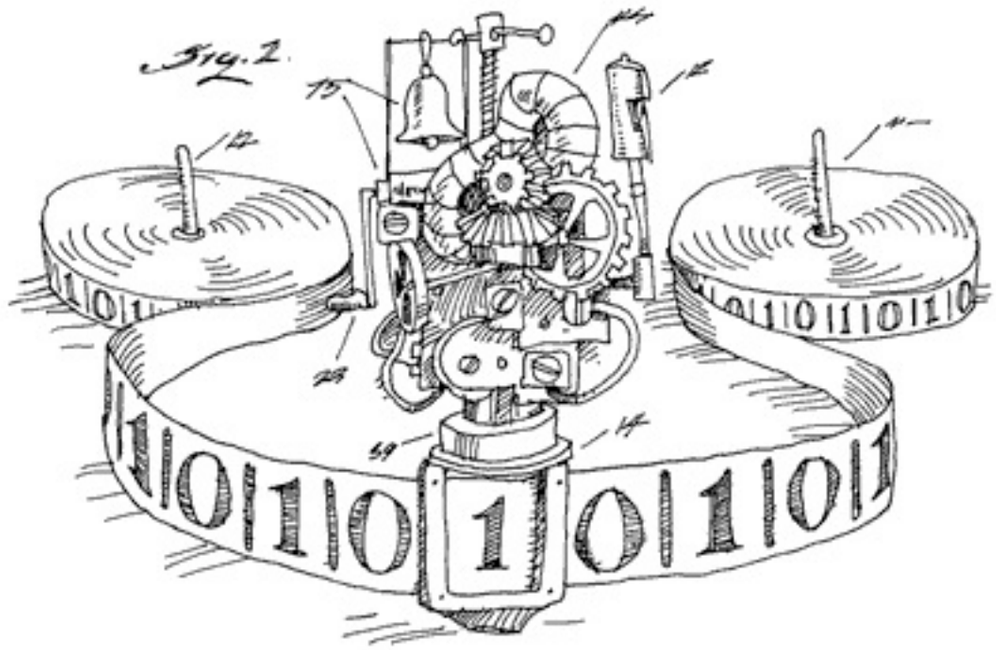
- MAS, a máquina de Turing...
  - é um modelo que define sistemas em termos de funções, e nem sempre esse é o tipo de abstração adequado;
  - é bastante difícil descrever sistemas não triviais em termos de máquinas de Turing.



Considerações

- MAS, a máquina de Turing...
  - é um modelo que define sistemas em termos de funções e nem sempre esse é o tipo de abstração adequado;
  - é bastante difícil descrever sistemas não triviais em termos de máquinas de Turing.

Sistemas computacionais evoluíram:
Outros modelos se tornaram necessários...

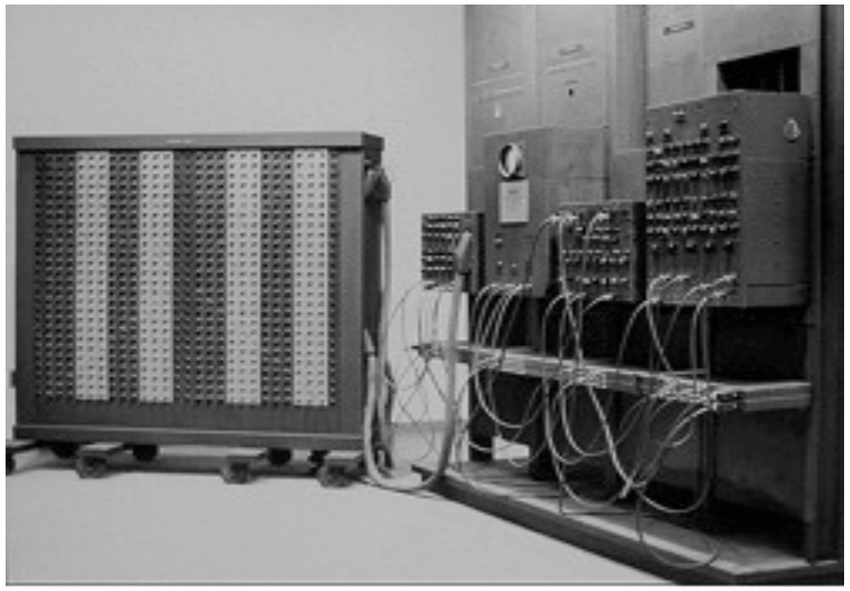


Evolução da Computação

- Contexto
- Técnicas de Engenharia de Software
- Teoria da computação: Métodos formais (linguagens, modelos semânticos, técnicas de análise, ...)

Evolução da Computação

Décadas de 1950-1960



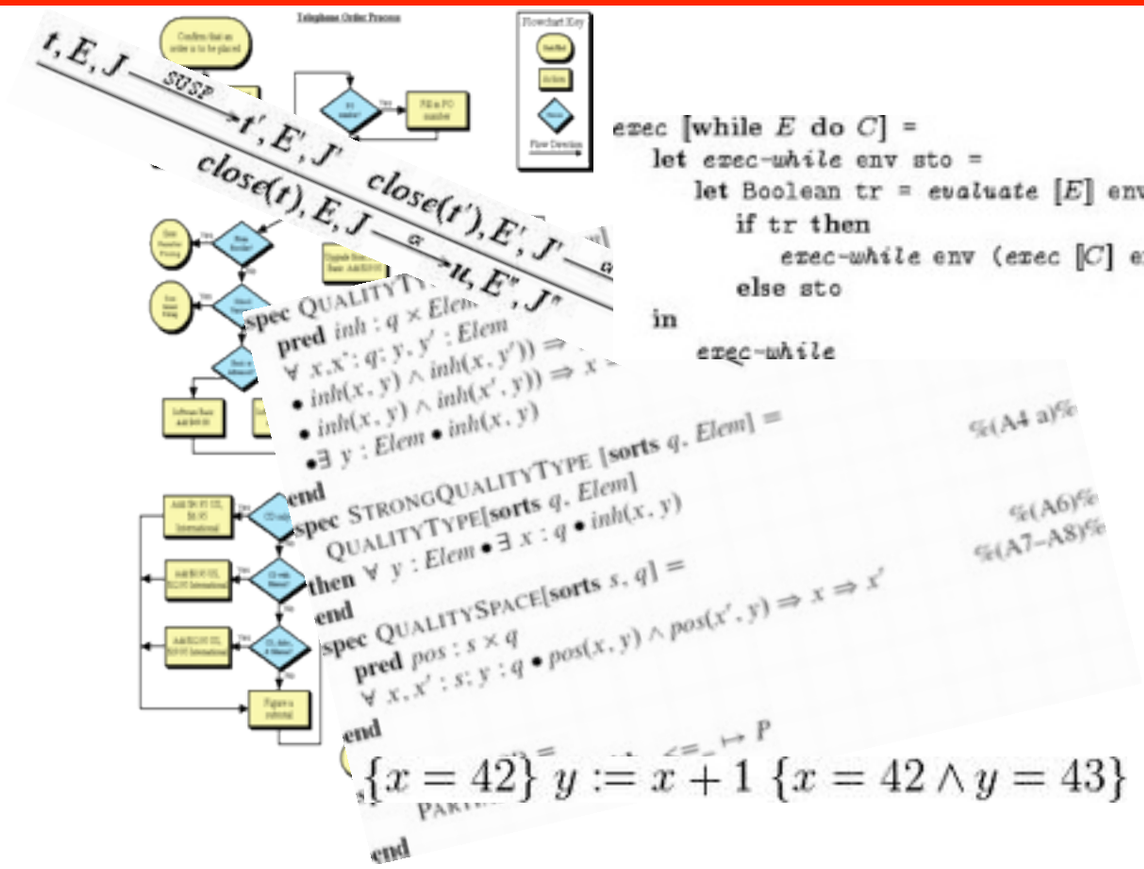
The ENIAC Today

- Programas eram muito simples: basicamente computavam funções
- Inicialmente a "*programação*" era manual: software e hardware se confundiam
- Mais tarde, "*programa armazenado*": primeira camada separando a solução do problema (SW) da máquina (HW) - sistema operacional
- Primeiras linguagens de programação: compiladores transformavam os programas em código executável em máquinas
- Primeiros modelos de computação: máquina de Turing, funções recursivas, máquinas de registradores, algoritmos de Markov, ..

Evolução da Computação

Décadas de 1970-1980:

- Complexidade do software aumenta
- Pouca independência do hardware
- Crise do software...

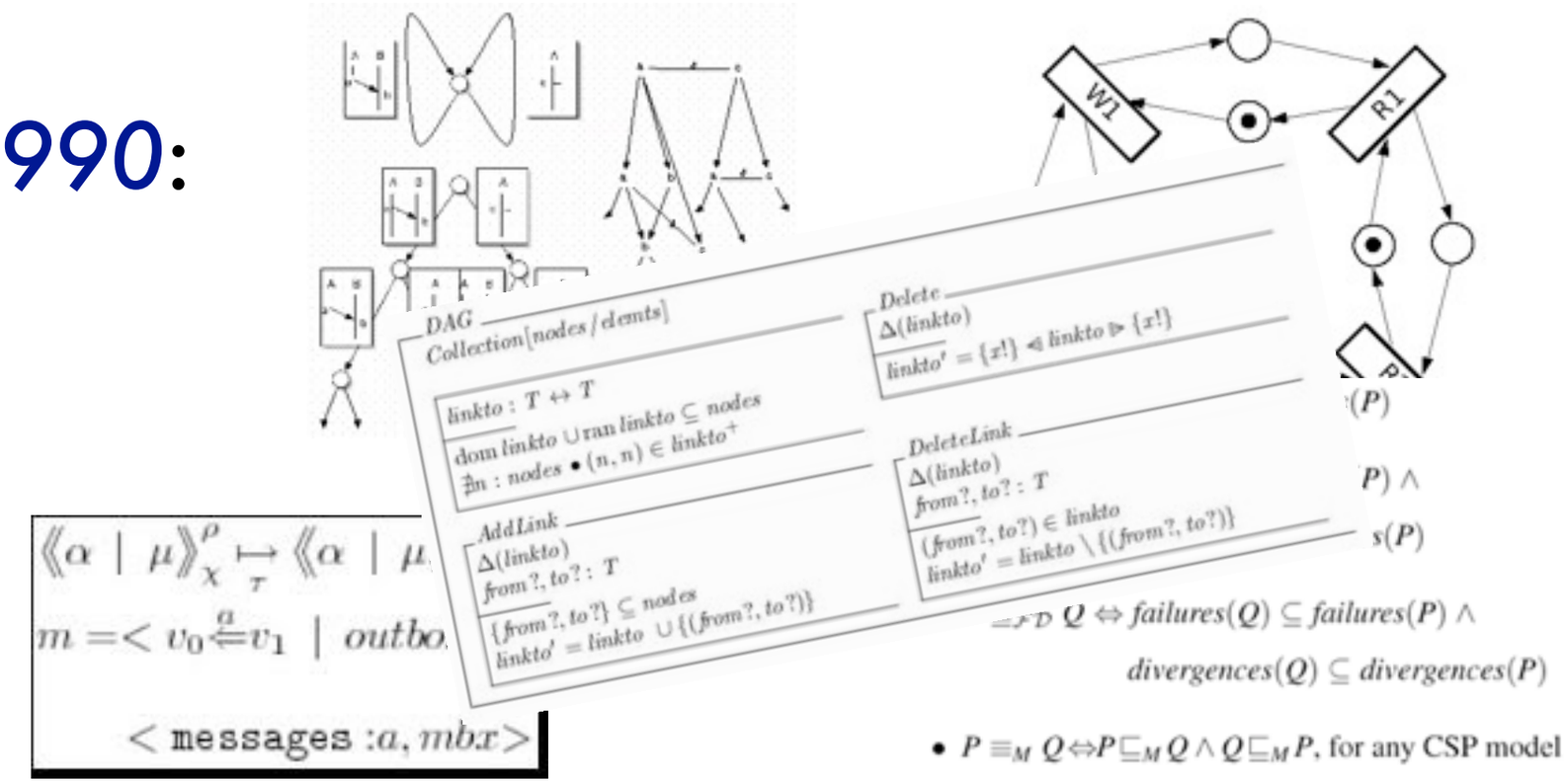


- Novas linguagens...
- Novas técnicas para construir programas: modularização, refinamento, orientação a objetos, ...
- Métodos formais para software: técnicas para descrever e analisar modelos, por exemplo especificação algébrica, VDM, triplas de Hoare, semântica denotacional, semântica operacional estruturada, ...

Evolução da Computação

Décadas de 1980-1990:

- Sistemas concorrentes

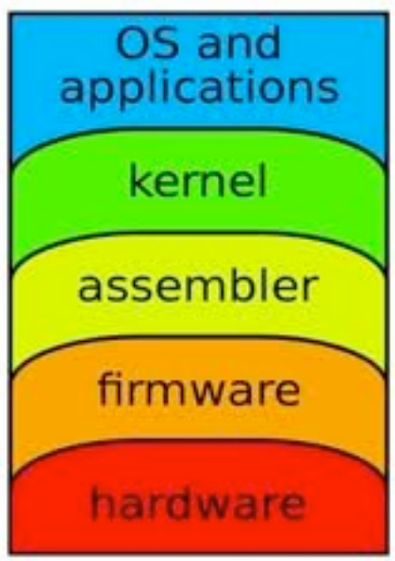


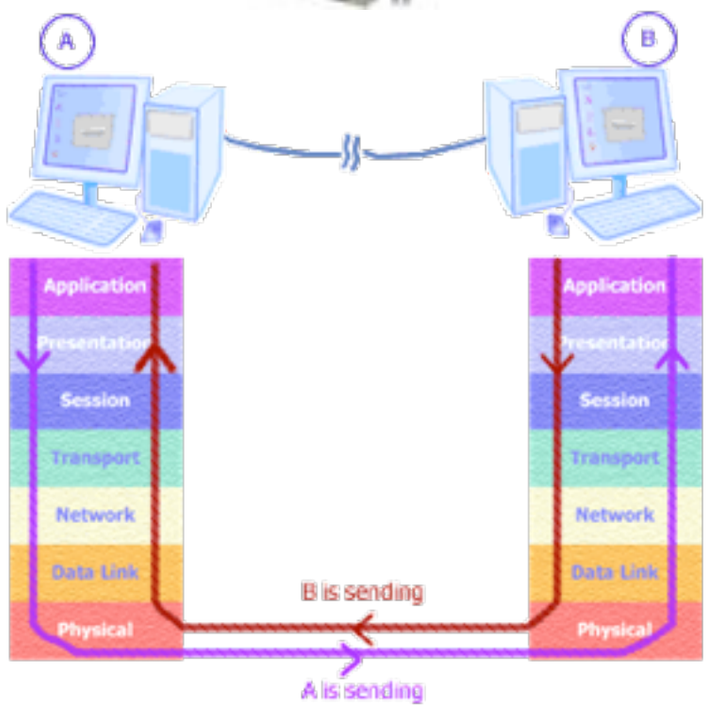
- Novos modelos: álgebras de processos, redes de Petri, Z, estruturas de eventos, traces, rewriting logics,...
- Grandes discussões sobre semântica de concorrência (true concurrency X interleaving, branching structures X linear structures, ...)

Evolução da Computação

Década de 1990-2000:

- Complexidade do software aumenta ainda mais
- Sistemas baseados em componentes
- Redes de computadores: comunicação, falhas, ...
- Mais independência do hardware



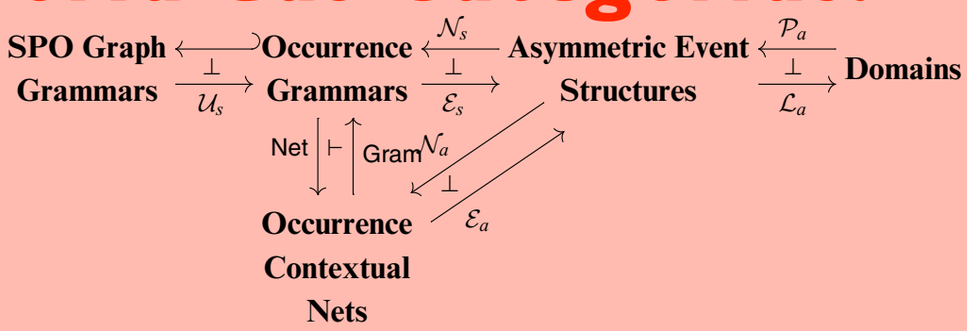


Evolução da Computação

Década de 1990-2000:

- Metodologias OO (OMT, OOD, ...), ferramentas CASE, UML
- Ferramentas para métodos formais
- Novos modelos: novas álgebras de processos, event-B, ASM, CASL, ...
- Novas técnicas formais para relacionar modelos:

Teoria das Categorias:



Instituições:

$$\mathbf{Mod}: \mathbf{Sign} \rightarrow \mathbf{Cat}^{op}$$

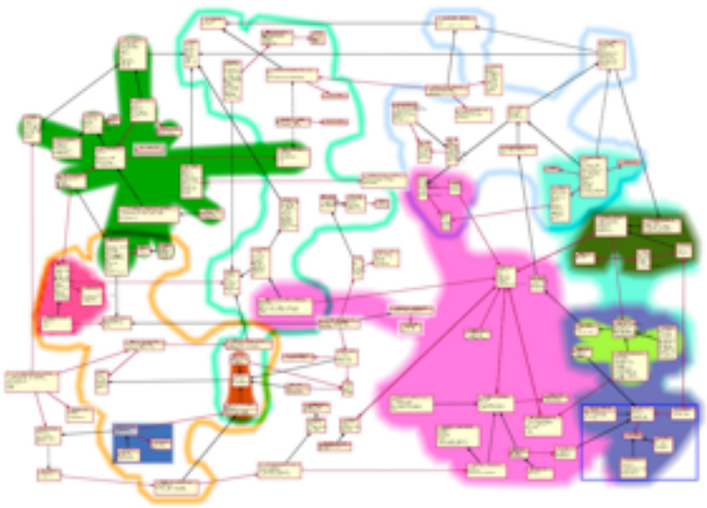
$$m' \models_{\Sigma'} Sen(\phi)(e) \quad \text{iff} \quad \mathbf{Mod}(\phi)(m') \models_{\Sigma} e$$

Truth is invariant under change of notation.

Evolução da Computação

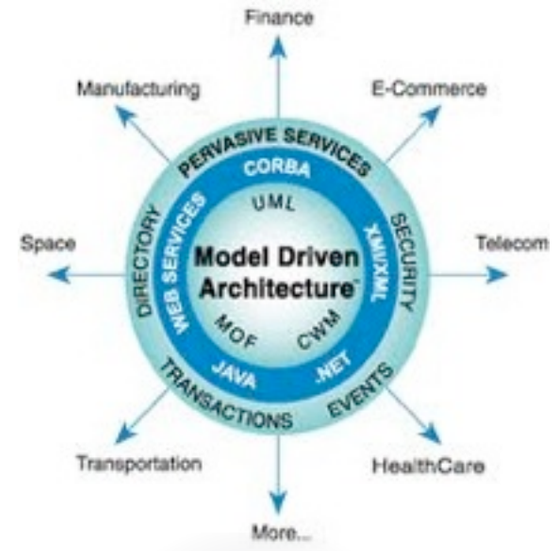
A partir de 2000...:

- Sistemas mais complexos:
  - * sistemas cooperativos
  - * sistemas muito heterogêneos
  - * sistemas embarcados
  - * paralelismo maciço
  - * mobilidade (software, hardware)
  - * falhas
- Preocupação com a evolução dos sistemas (novos requisitos, novas plataformas, novas tecnologias, ...)

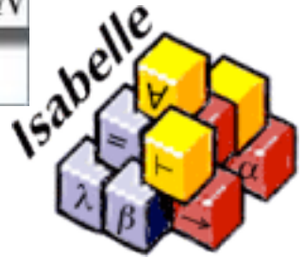


Evolução da Computação

A partir de 2000...:



- Uso em larga escala de modelos padronizados (por exemplo, UML da OMG)
- Model Driven Engineering (MDE)
- Maior preocupação com modelos e como realizar transformações entre modelos
- Grande número de métodos para análise de modelos computacionais: verificadores de modelos, provadores de teoremas, análise quantitativa, análise estática, ...
- Maior maturidade de ferramentas que apoiam métodos formais.

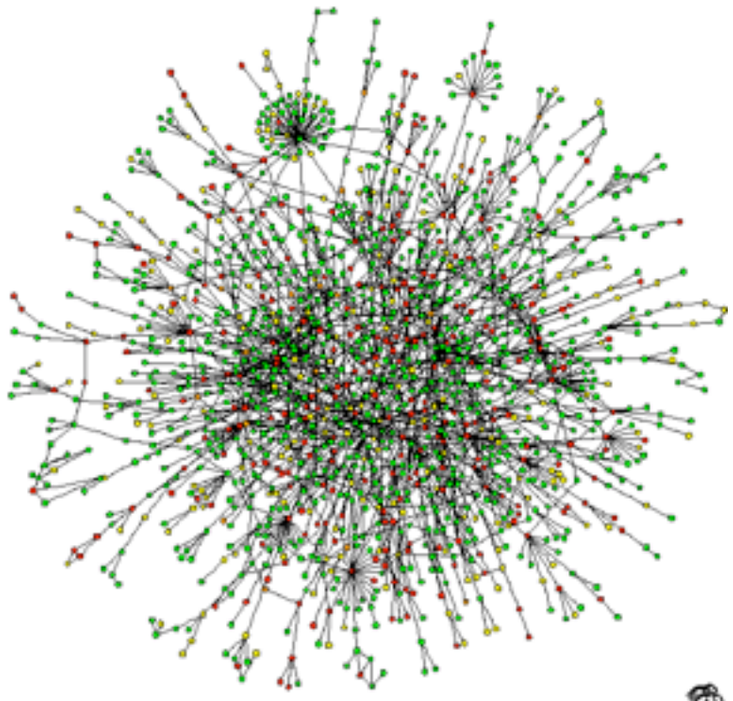


Hets

Maude

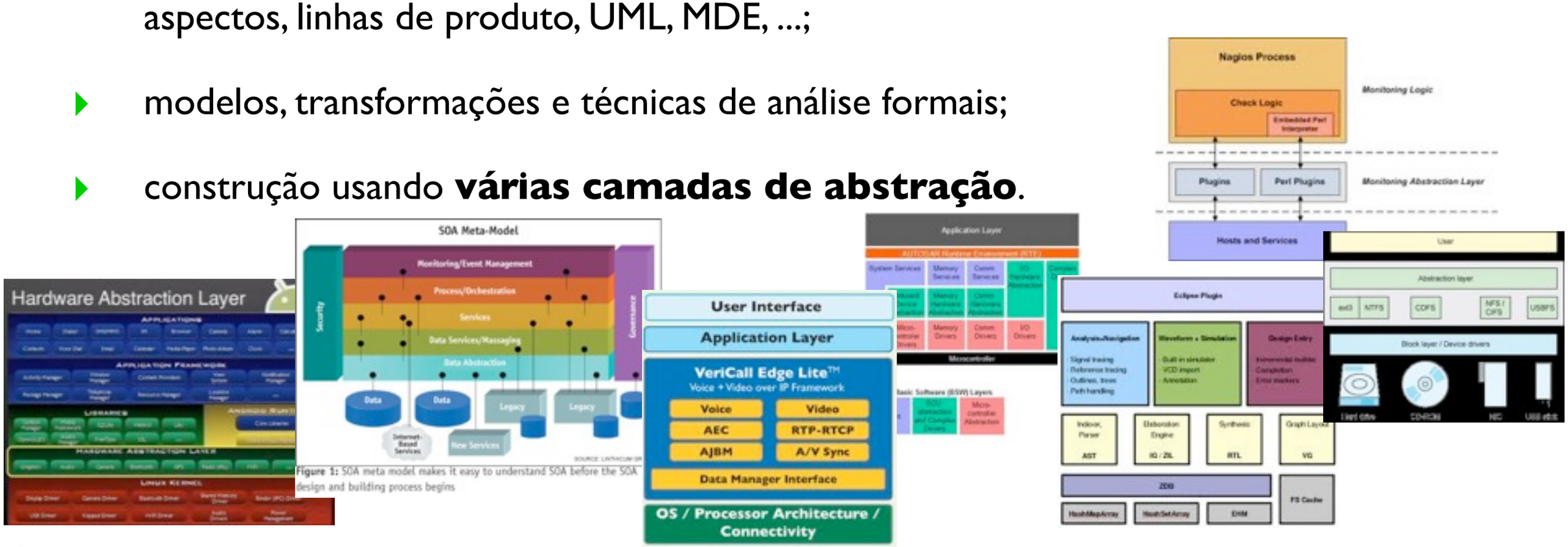
Voltando ao assunto...

- Que problemas podem ter suas soluções e implementadas em "máquinas inteligentes e obedientes"???
  - Inicialmente, problemas cujas soluções são funções (computáveis)
  - Hoje, problemas envolvendo sistemas complexos, caracterizados por
    - número muito grande de componentes;
    - heterogeneidade;
    - comportamento emerge da comunicação/reação;
    - mobilidade;
    - falhas;
    - evolução



Voltando ao assunto...

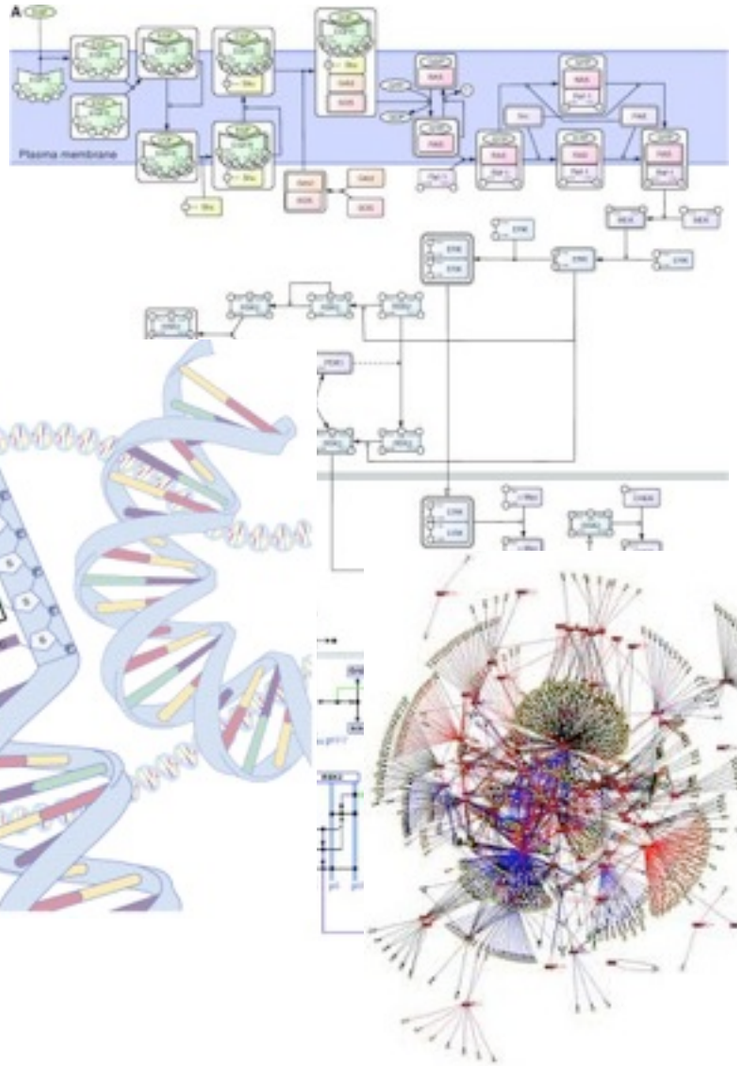
- Como construir as soluções???
  - Inicialmente, sequências de passos bem definidos
  - Hoje, utiliza-se várias técnicas para tratar a complexidade do sistema
    - metodologias de Engenharia de Software: modularização, refinamentos, padrões, aspectos, linhas de produto, UML, MDE, ...;
    - modelos, transformações e técnicas de análise formais;
    - construção usando **várias camadas de abstração**.

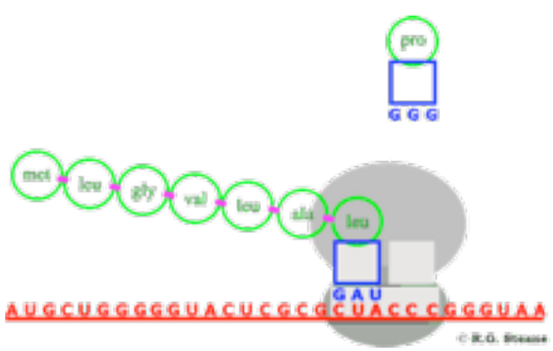


Generalizando...

- Os modelos desenvolvidos para software "tradicional" podem ser usados também para descrever **outros sistemas complexos**.

- Por exemplo, podemos usar modelos e técnicas desenvolvidas pela **Computação** para descrevar, analisar e entender Sistemas Biológicos, ou seja, para entender **como a natureza computa.**





Pensamento Computacional

"Computational thinking is the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information processing agent." [CunySnyderWing10]

Jeannette Wing

- Pensamento computational é:
  - pensar recursivamente;
  - reformular um problema complexo para um que sabemos resolver (redução, transformação, simulação);
  - escolher a melhor representação dos aspectos relevantes de um problema para torná-lo tratável;
  - interpretar dados como programas e programas como dados;
  - usar composição e decomposição para tratar tarefas grandes;
  - pensar em como/quando paralelizar e como/quando sincronizar tarefas; ...

Pensamento Computacional

"Computational thinking is the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information processing agent." [CunySnyderWing10]

Jeannette Wing

- Pensamento computational é:

...usar uma abordagem para resolver problemas, projetar sistemas e entender o comportamento humano que tem como base os fundamentos da Ciência da Computação!

Pensamento Computacional

- Pensamento computational pode ser colocado como uma das habilidades intelectuais básicas de um ser humano, comparada à ler, escrever, falar e fazer operações aritméticas.
- Essas habilidades fundamentais servem para descrever e explicar situações complexas.
- Nesta linha de raciocício, o Pensamento Computacional é mais uma linguagem (junto com as linguagens escrita e falada, e a matemática) que podemos usar para falar sobre o universo e seus processos complexos.





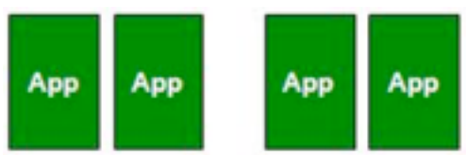
Exemplos Simples do Cotidiano

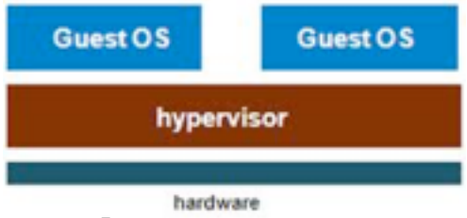
- Montar um roteiro de viagem (caixeiro viajante)
- Procurar um nome em uma lista ordenada (linear search/binary search)
- Arrumar a mala (problema da mochila)

- Organizar um jantar para 30 pessoas
- Ordenar pilhas de figurinhas

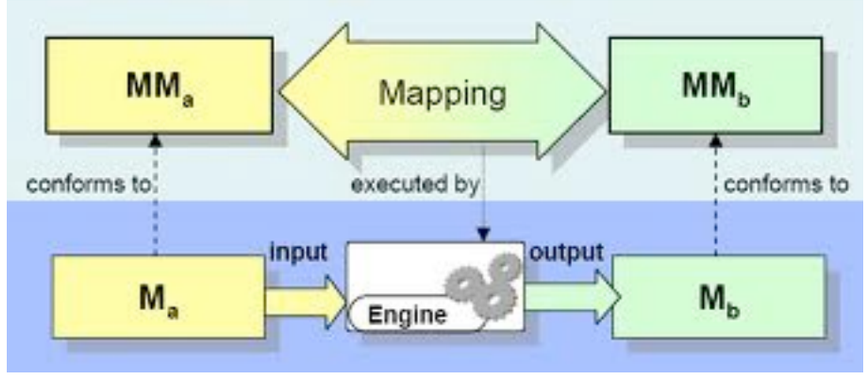
Pensamento Computacional

- Um dos pilares do Pensamento Computacional é a **Abstração**: Dada a complexidade dos problemas, são necessários vários modelos diferentes para descrevê-los, tanto
  - **horizontalmente (diferentes aspectos)** quanto
  - **verticalmente (diferentes níveis de abstração).**





- São necessários formas bem definidas para relacionar estes diferentes modelos (através de vários tipos de transformações)

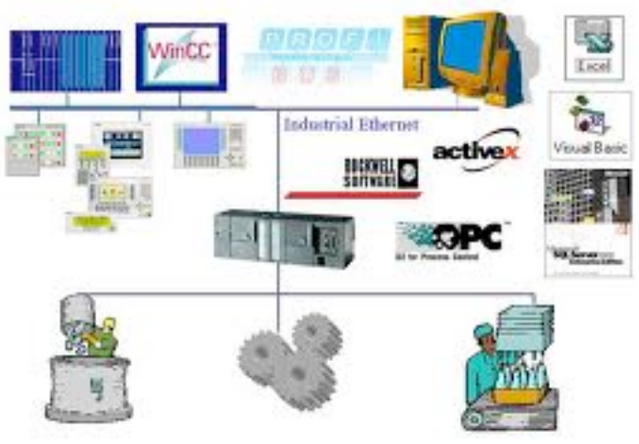


Pensamento Computacional

- O outro pilar do Pensamento Computacional é a **Automação**: Automatizar é mecanizar as abstrações, e as relações entre níveis de abstrações.

- Mecanização é possível se associamos significados precisos às abstrações

- A automação permite que alguma "máquina" auxilie na solução dos problemas.



Habilidades

- Pensamento computacional significa
  - entender quais aspectos de um problema podem ser resolvidos usando Computação;
  - avaliar e escolher as ferramentas e técnicas computacionais adequadas para cada problema;
  - entender as limitações das ferramentas e técnicas computacionais;
  - reconhecer oportunidades de usar Computação de novas maneiras;
  - aplicar estratégias computacionais, como divisão e conquista, em qualquer domínio

Habilidades

- Pensamento computacional para cientistas e engenheiros significa
  - aplicar novos métodos computacionais aos problemas;
  - reformular problemas para permitir a aplicação de técnicas computacionais;
  - fazer novas descobertas através da análise de grandes volumes de dados;
  - fazer novas perguntas que não eram sequer cogitadas pelas dificuldades de escalabilidade que suas respostas necessitam;
  - explicar problemas e soluções em termos computacionais.

Links

- http://www.cs.cmu.edu/~CompThink/
- http://www.google.com/edu/computational-thinking/index.html
- http://csunplugged.org/
- Computational Thinking: A Digital Age Skill for Everyone - YouTube

Pensamento Computacional

- Escolher as abstrações certas
- Escolher o "computador" ideal para as tarefas

O que é Computação?

- Como resolver problemas?

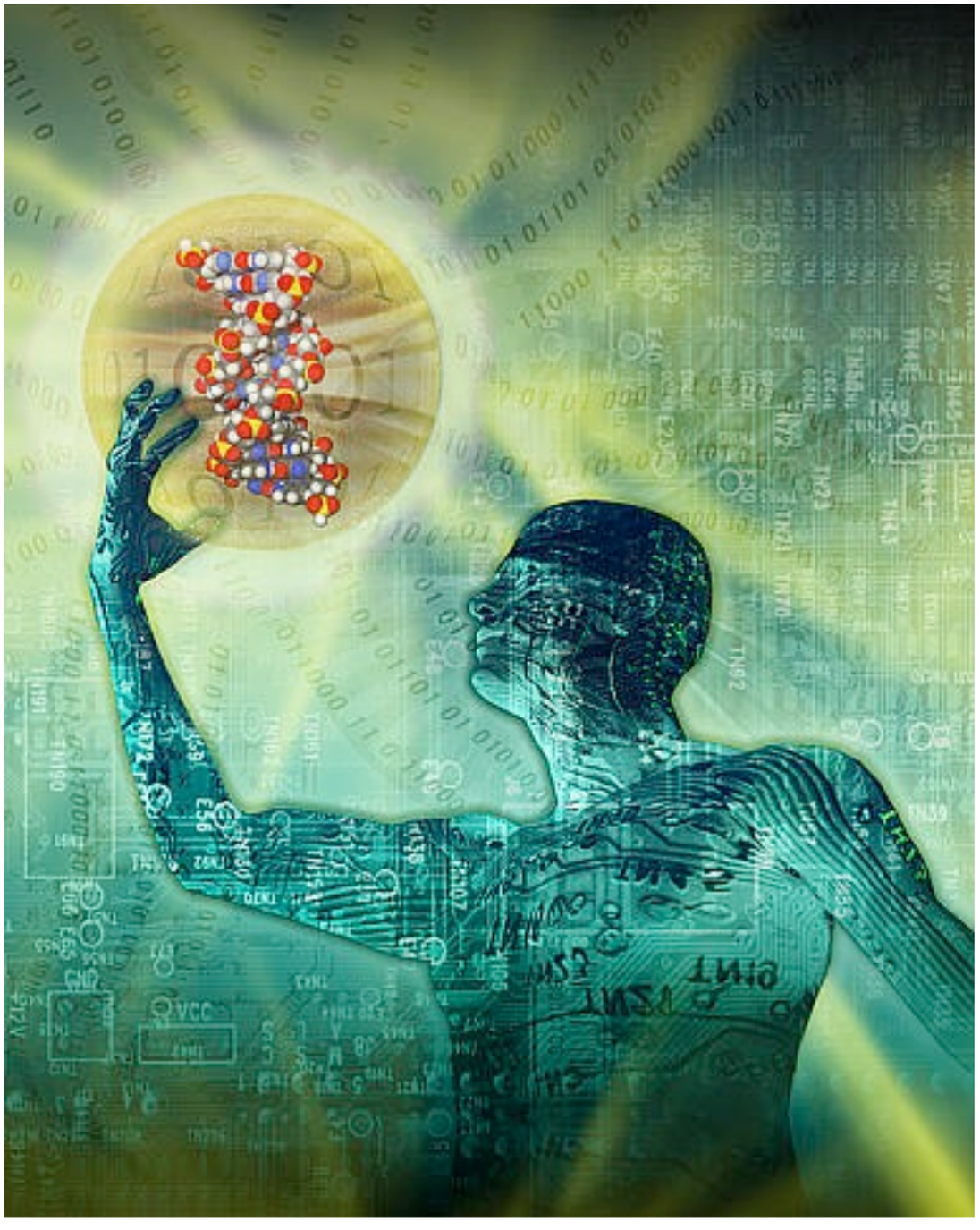
- Como fazer com que "máquinas" nos ajudem a resolver problemas?

O que é Computação?

Pensamento Computacional

- Como resolver problemas?
  - Escolher as abstrações certas
- Como fazer com que "máquinas" nos ajudem a resolver problemas?
  - Escolher o "computador" ideal para as tarefas

O que é Computação?



O trabalho iniciado por Alan Turing continua...

A área de Computação pode ajudar a Humanidade a entender/projetar sistemas extremamente complexos e assim atingir patamares de conhecimento muito mais elevados!