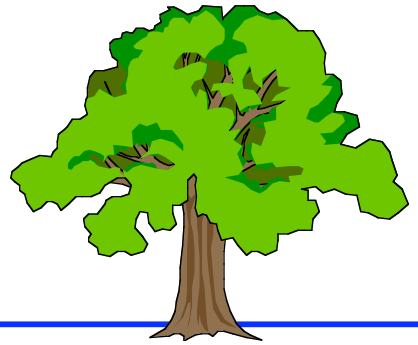
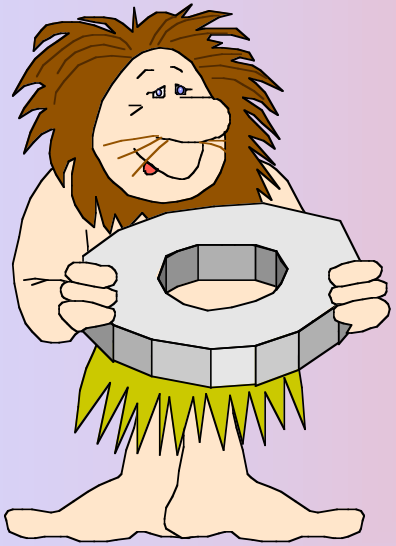
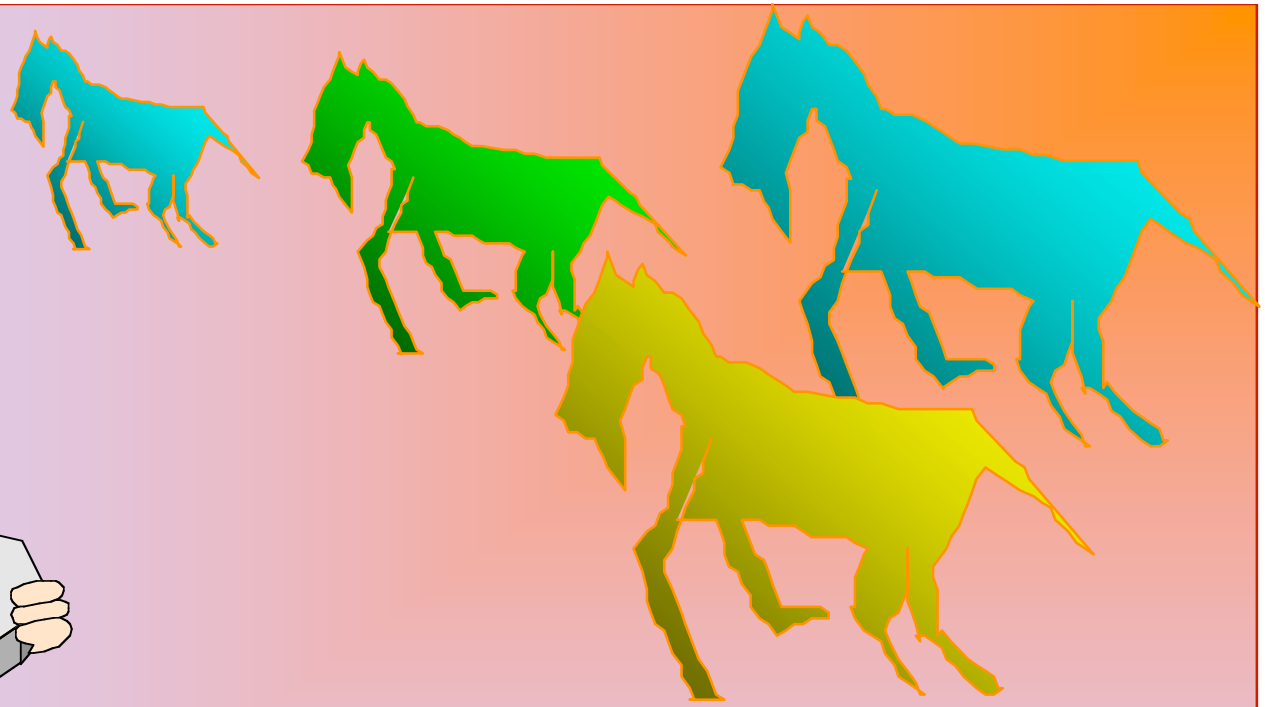
I. Introduction

What is Modeling?
Impressions and Expressions
History of Conceptual Modelling
Conceptual Models
Primitive Terms, Abstraction Mechanisms and Tools
EER, UML, CLASSIC, KAOS, Telos, …,and Tropos
Formality for Modeling Languages







Humans used symbols to model their environment since the beginning of civilization!

5,000 - 10,000 Year Ago …

Prehistory of Modeling

→ … And they kept doing it through Art, Science and Engineering.

→ But their models were bounded by the size of the medium on which they were represented* …

… until computers came along …

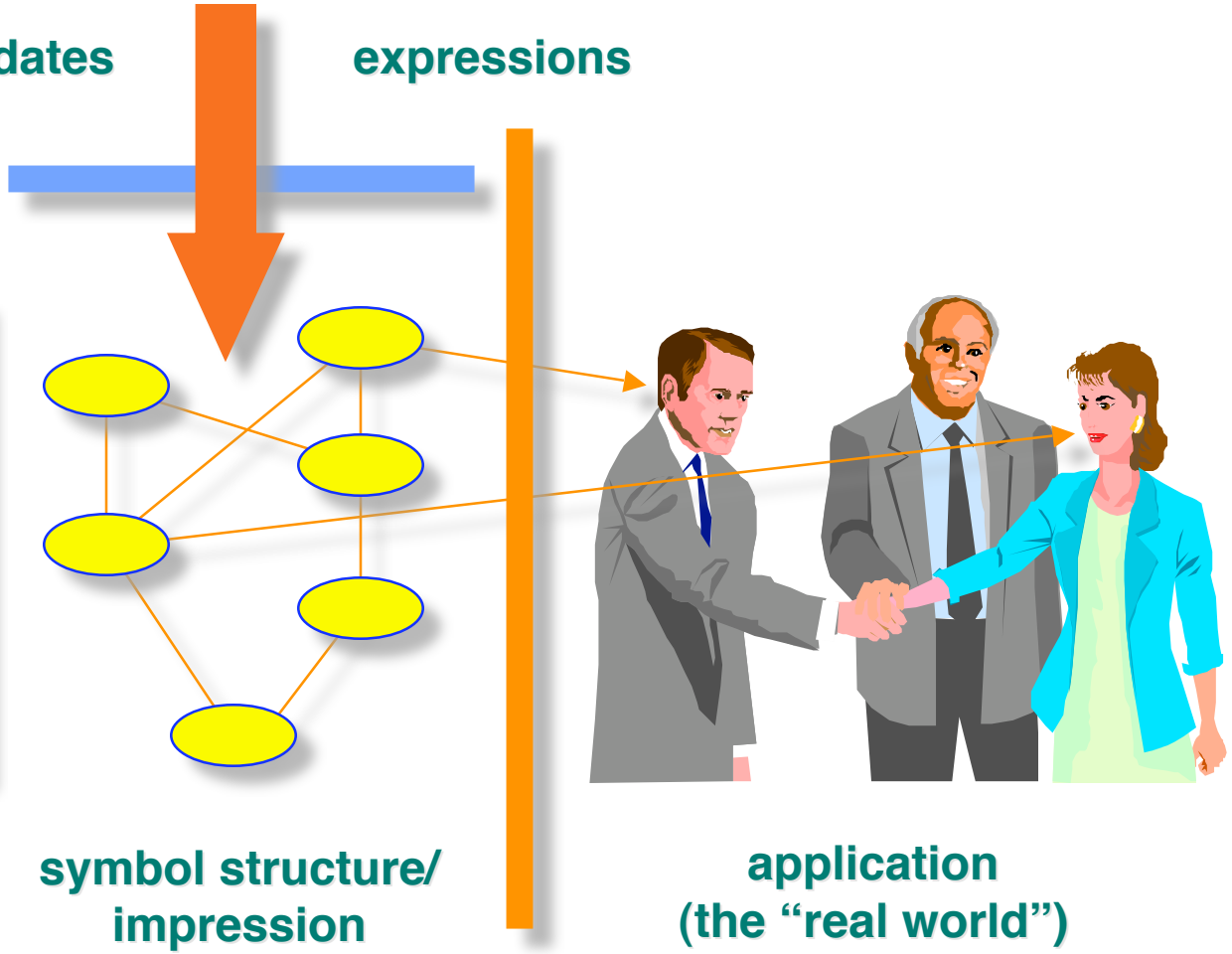
→ Suddenly, the limits of models and modeling were the same with the physical limits of the machines on which they were represented.

→ The rest of the story of modeling is intertwined with that of Computer Science.

* Check out huge tapestry at Galleria degli Uffizi representing a battle.

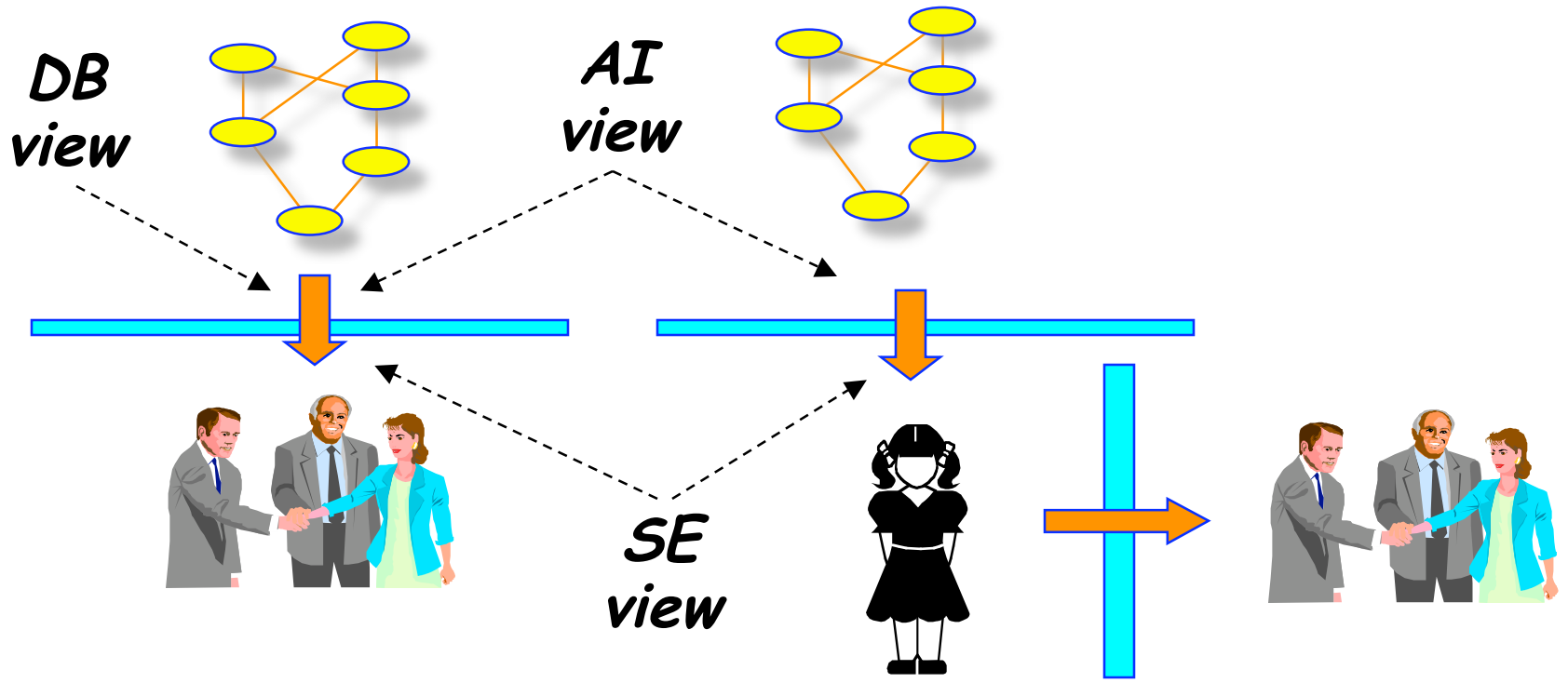
What is (Conceptual) Modeling?

The use of **computer-based symbol structures (impressions)** to model (somebody's conception of) an application



...Actually...

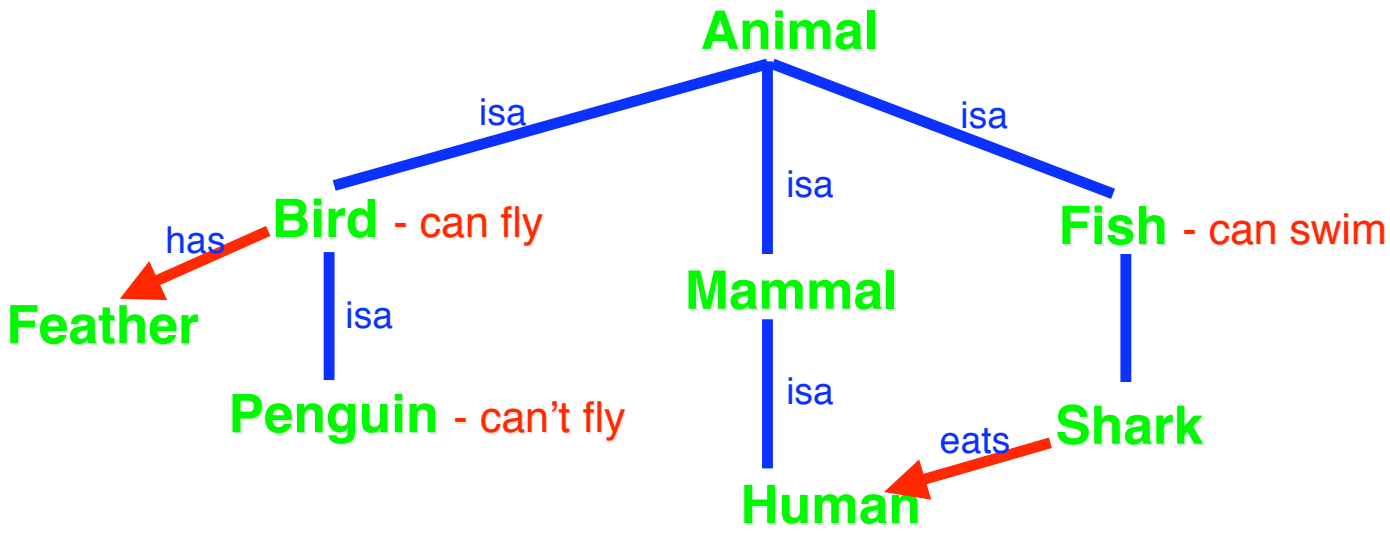
We could be modeling the real world, or *somebody's conception* of the real world



Origins in Computer Science

- Ross Quillian proposed *semantic networks* as a model of the structure of human memory (1966).
- Ole-Johan Dahl proposed in 1967 *Simula*, an extension of the programming language ALGOL 60, for simulation applications which require some "world modeling"
- Jean-Raymond Abrial proposed a *semantic model* in 1974, shortly followed by Peter Chen's *Entity-Relationship model* (1975) as advances over logical data models, such as Codd's Relational model.
- Doug Ross proposed in the mid-70s the *Structured Analysis and Design Technique* (*SADT*) as a "language for communicating ideas". The technique was used to specify requirements for software systems.

Semantic Networks (1966)



Novel ideas

→ Impressions are built out of *concepts* and *associations*

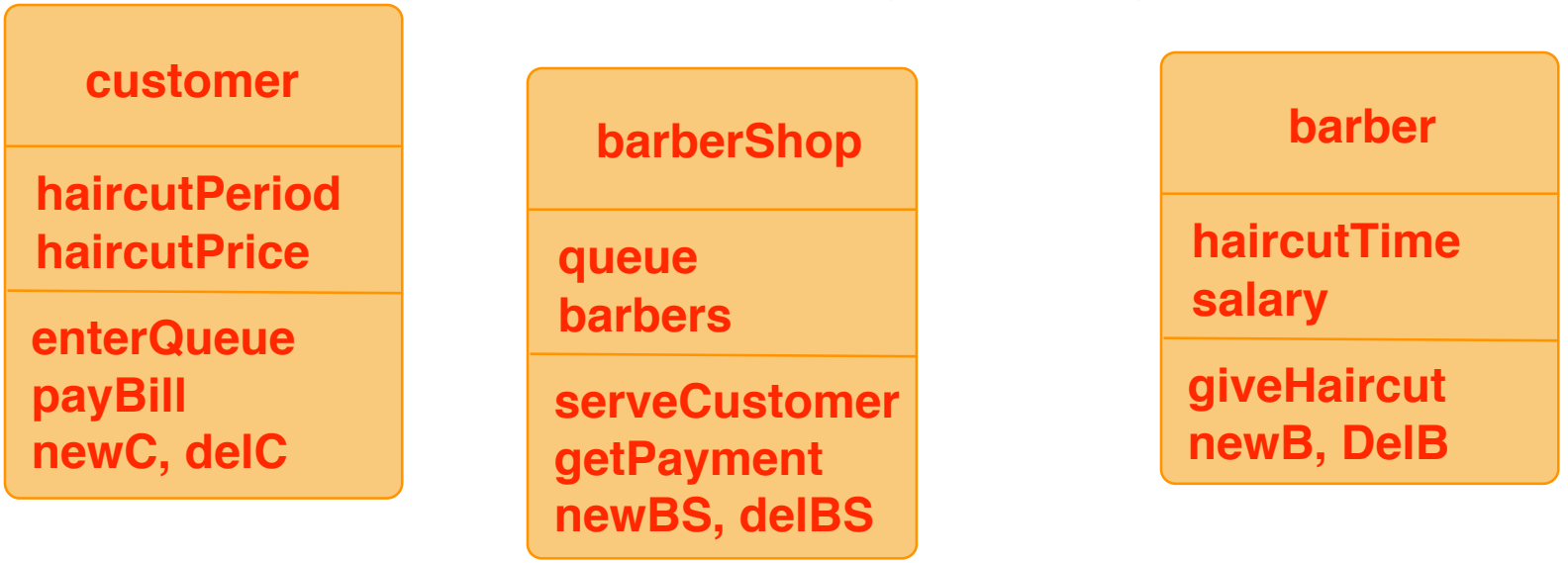
→ *Inheritance of attributes* -- default, single

→ Computation defined in terms of *spreading activation* -- e.g., discovering the meaning of "horse food"

horse --> animal --> eat --> food

horse --> animal --> madeOf --> meat --> food

Simula (1967)

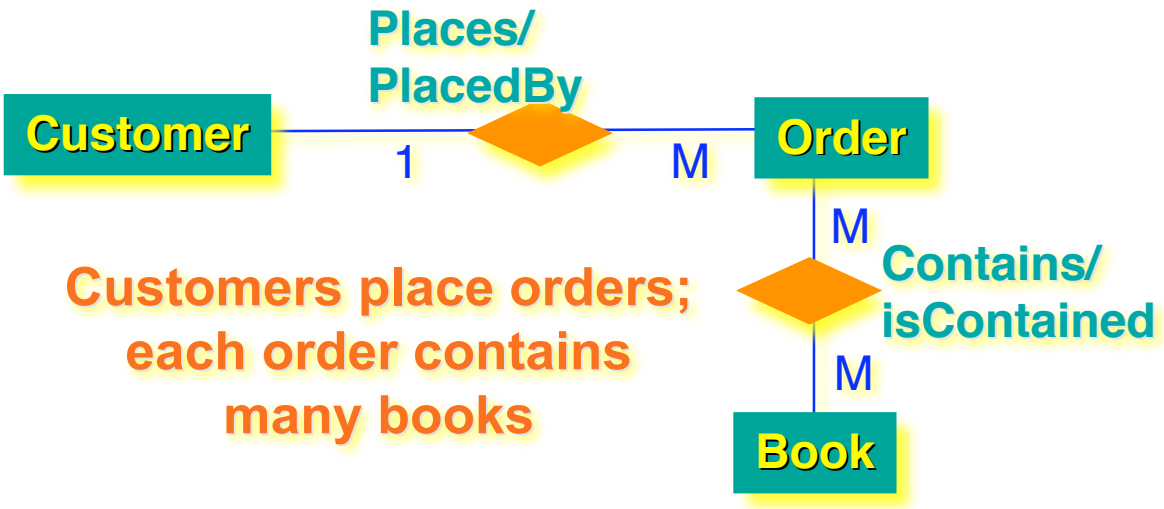


→ Ole-Johan Dahl proposed it as an extension of the *ALGOL 60*, for simulation applications.

→ *A* (simulation) program consists of classes and instances.

→ Instances *model the simulated application*, classes define common features of instances, are organized into subclass hierarchies.

The Entity-Relationship Model (1975)



Novel ideas

→ Assumes that application consists of *entities* and *relationships* (*ontological assumptions*)

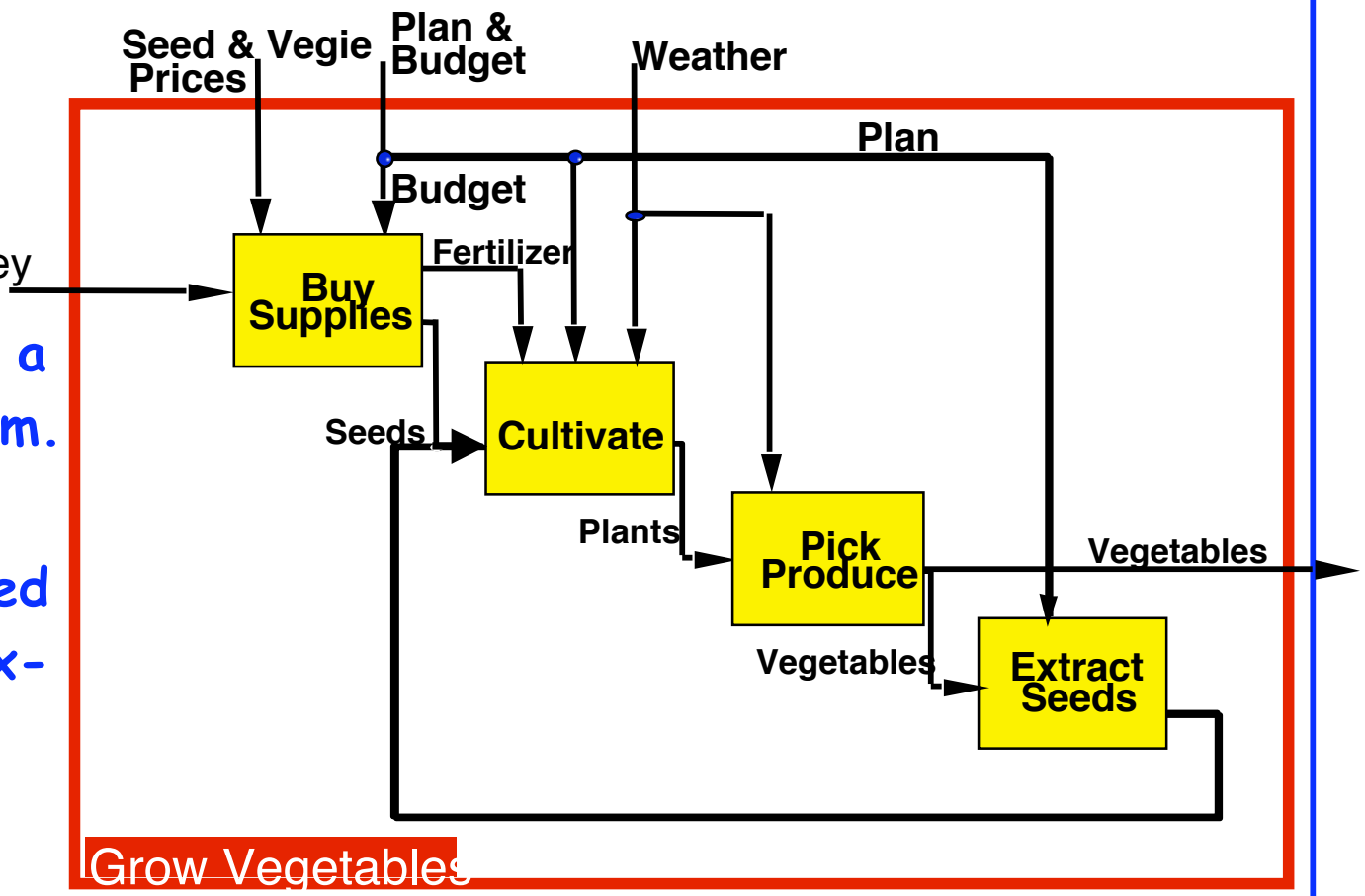
→ Shows how a conceptual schema can be mapped onto a logical one.

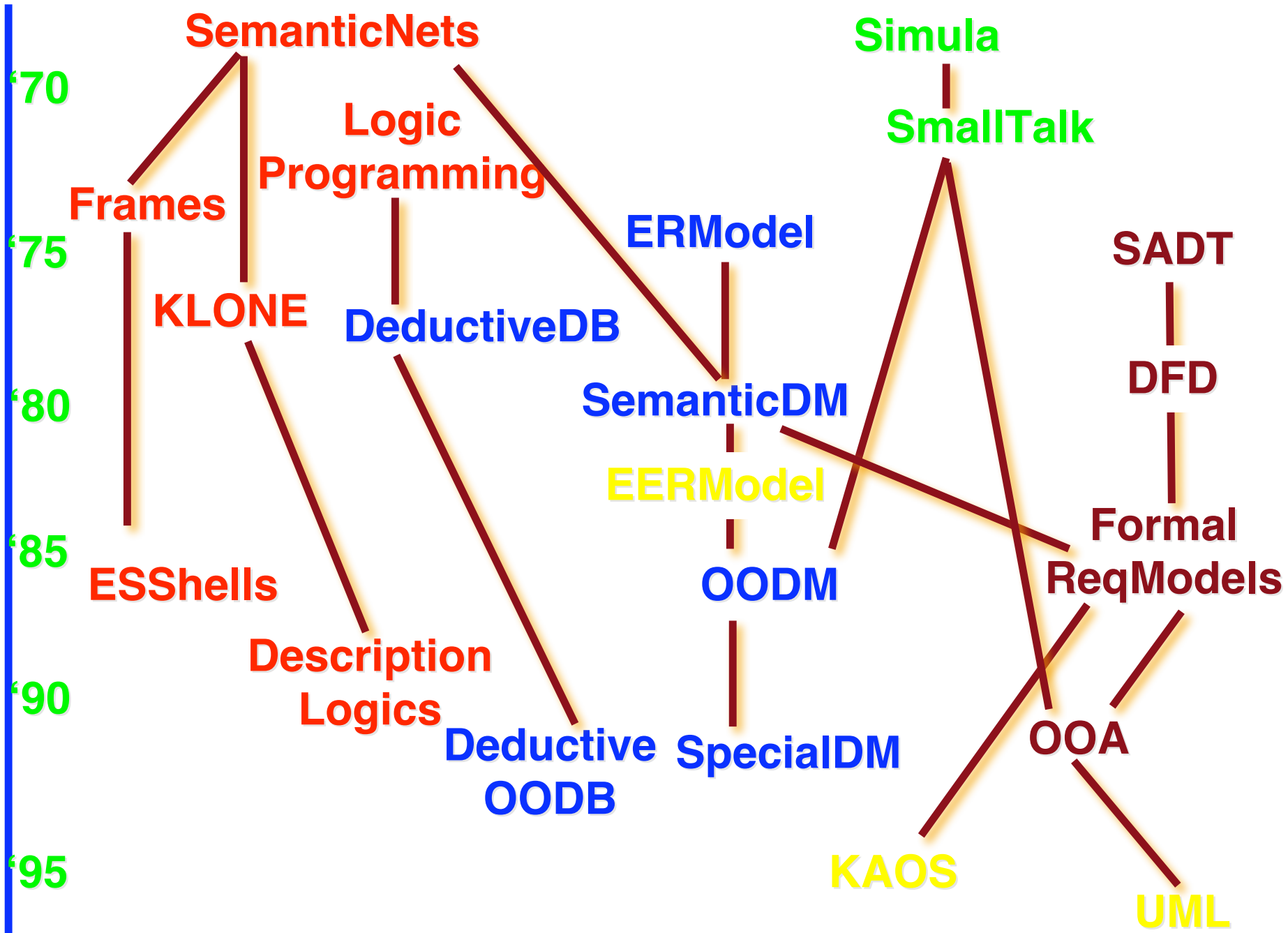
→ [Abrial's semantic model was more akin to OO data models, but did offer entities and relations too]

Structured Analysis and Design Technique (SADT)

Novel Ideas

- → Modeling the operating environment of a software system.
- → Application models organized in terms of box-inside-box notation.





Different areas of Computer Science use different terms for the same concepts!

Expressions

→ *Expressions* are communication statements of linguistic and/or pictorial nature through which an agent (human or otherwise) accumulates information about an application.

- ✓ Natural language statements are expressions.
- ✓ So are books...
- ✓ So are Predicate Calculus statements...
- ✓ So are SQL statements....
- ✓ So are digitized pictures...
- ✓ So are Web pages...
- ✓ ...more...

Impressions

- *Impressions* (also *models*) are internal symbol structures held by an agent (human/software) that represent fragments of an application.
- Example impressions: human memory, mental states, databases, knowledge bases, websites,...
- Impressions have two properties:
  - ✓ *Functional role* -- impressions arise by virtue of the history and coupling of the agent to its environment, and give rise to the agent's future actions;
  - ✓ *Representational import* -- impressions stand in a content relationship to an application the agent is in contact with.

Meaning vs Interpretation

- The *meaning* (*sense*) of an impression is what all uses of the impression have in common. For example, "the morning star" always refers to a star visible early in the morning.
- The *interpretation* (*reference*) of an impression is what the impression refers to in a particular instance, or a particular use. For example, "the morning star" may refer to planet Venus when I use it (...on Earth), while it refers to planet Jupiter when an astronaut says it on Mars.
- Typically, the transition from meaning to interpretation occurs when an impression is bounded to a context. This applies to indexical terms, e.g., "I", "here", "now", but also descriptions, such as "morning star" and "a student in CSC2507".

Analysis of Impressions

- Formality is important for a modeling language for two reasons:
  - ✓ It eliminates, or at least reduces, ambiguity;
  - ✓ It can be used as a foundation for (algorithmic) analysis of impressions.
- Analysis may consist of various forms of consistency checking, completeness checking, etc.
- The course will be discussing different forms of analysis for different kinds of impressions.

We don't have deep theories of impressions, yet!

Towards Theories of Impressions

- → Psychologists since Aristotle have offered *associationism* as a theory of how the brain is organized [Anderson]
- → McCarthy proposed *abstract syntax trees* in the early 60s.
- → Linguists have talked about the *deep structure of language.*

Modeling in Computer Science

Modeling has been practiced and researched in several areas within Computer Science and Engineering, including:

→ **Databases** -- data modeling, using vanilla or semantic data models to build *databases*;

→ **Software Engineering** -- requirements modeling, using diagrammatic (structured, OO, or plain vanilla) techniques to build *requirement specifications*; software process modeling using finite state machines, statecharts, rules, Petri nets to build *process models*;

→ **Artificial Intelligence** -- knowledge representation using logic-based notations, description logics, semantic networks, frames, etc. to build *knowledge bases*;

Variations in Use

- There are important differences in the use of impressions among different areas of Computer Science:
- In AI, knowledge bases are to be used by *a program* to perform a task (expertly).
- In Software Engineering, diagrams are used for communication *among people*.
- Databases model large amounts of concrete facts, rather than generic knowledge, are used by *humans and programs* and have to scale up.

Implications

Differing uses of impressions have important consequences

→ *Type of notation used* -- if people use the impression, the notation may be semi-formal or even informal; for knowledge bases in AI, on the other hand, formality is all-important.

→ *Types of knowledge captured* -- for SE and Databases domain information is captured; for AI, task knowledge, including heuristics is equally important.

→ *Coverage and completeness of the impressions* -- for SE and Databases coverage can be quite broad and incomplete; for AI, coverage has to be narrow and complete.

Modeling Today

- Widely used for design (databases, software), semantics (semantic web, digital libraries, …), model-driven architectures (MDA) and engineering (MDE), more.
- Two developments contributed greatly to this:
  - ✓ Establishment of a standard (UML, 1995)
  - ✓ Ontologies (Knowledge sharing initiative, 1991)

Types of Models

- *Physical models* use machine-oriented concepts -- e.g., records, fields, B-trees,...-- to model an application.

  ...conflicting representational and efficiency concerns!

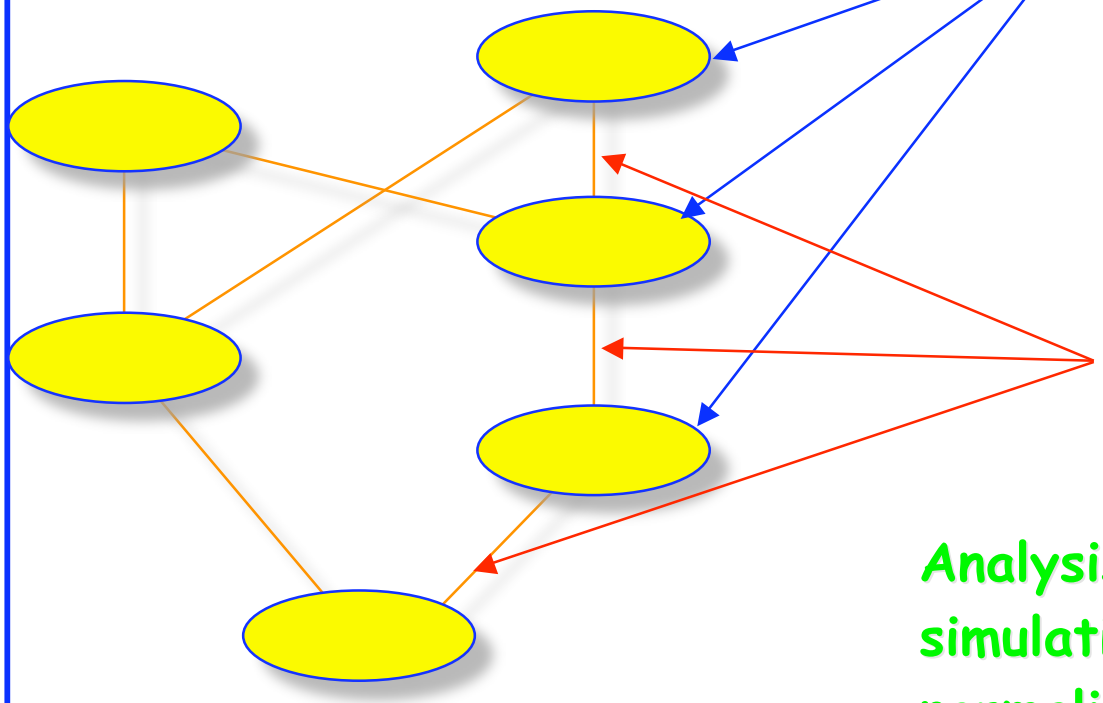
- *Logical models* offer mathematical abstractions -- e.g., arrays, lists, sets, relations -- for modeling purposes, hiding the implementation details from the user.

  ...but logical symbol structures are flat and unintuitive....
  can't describe complex situations!

- *Conceptual models* use application-oriented terms and organize information on the basis of principles such as generalization, aggregation and classification.

  conceptual symbol structures (are supposed to)
  model directly and naturally an application

Characterizing Modeling Languages



Basic building blocks -- Terms
descriptions, concepts,...

Composition rules --
Abstractions, algebra,
semantic relationships

Analysis and management tools
simulation, verification,
normalization,...

Terms, Abstractions, and Tools

→ Let's try to characterize conceptual models by looking at the basic building blocks, the structuring mechanisms, and the tools they offer for building an information base:

→ Primitive Terms - these are the concepts built into a conceptual model, e.g., Entity, Activity, Agent, Goal,...

→ Abstraction Mechanisms -- primitive mechanisms for structuring an impression along different dimensions, e.g., Generalization, Aggregation, Classification,...

→ Tools and Analysis techniques -- for creating, updating, searching, validating and managing an information base

Important to avoid superficial comparisons,

e.g., ones based on syntactic/graphical sugar

Three Modeling Languages

- Extended Entity-Relationship Model (EER) -- ER model extended to support aggregation and generalization; there are many different version of this, e.g., [EER92]; some versions are supported by commercial modeling tools.
- Unified Modeling Language (UML) -- combines earlier Object-Oriented Analysis (OOA) techniques by Booch, Jacobson (OOSE), Rumbaugh (OMT), others [UML97]; offers facilities for modeling objects, methods, various types of actors, aggregation, generalization, etc.
- CLASSIC -- A Description Logic developed at Bell labs by Brachman, Borgida etc. [Borgida89]; uses a limited form of Logic, supports a limited form of inference.

...And Three More...

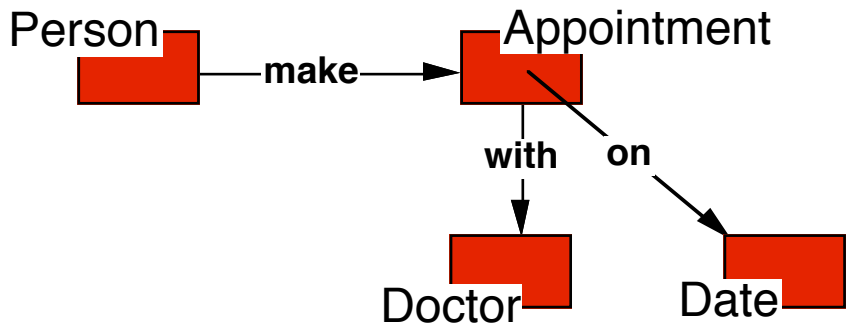
- KAOS -- a research prototype for requirements modeling, developed by Axel van Lamsweerde and colleagues [Dardenne93]. Can model goals, constraints, tokens, classes, metaclasses etc.
- Telos -- a language intended for metamodelling applications [Mylopoulos90]; treats attributes as first class objects, uses heavily metaclasses (for objects and attributes) for metamodelling, supports a logic-based sublanguage for specifying constraints and deductive rules; see also RDF, MOF.
- Tropos -- a modeling language founded on the notions of actor, goal, and social dependency (among actors); used for modeling different phases of software development.

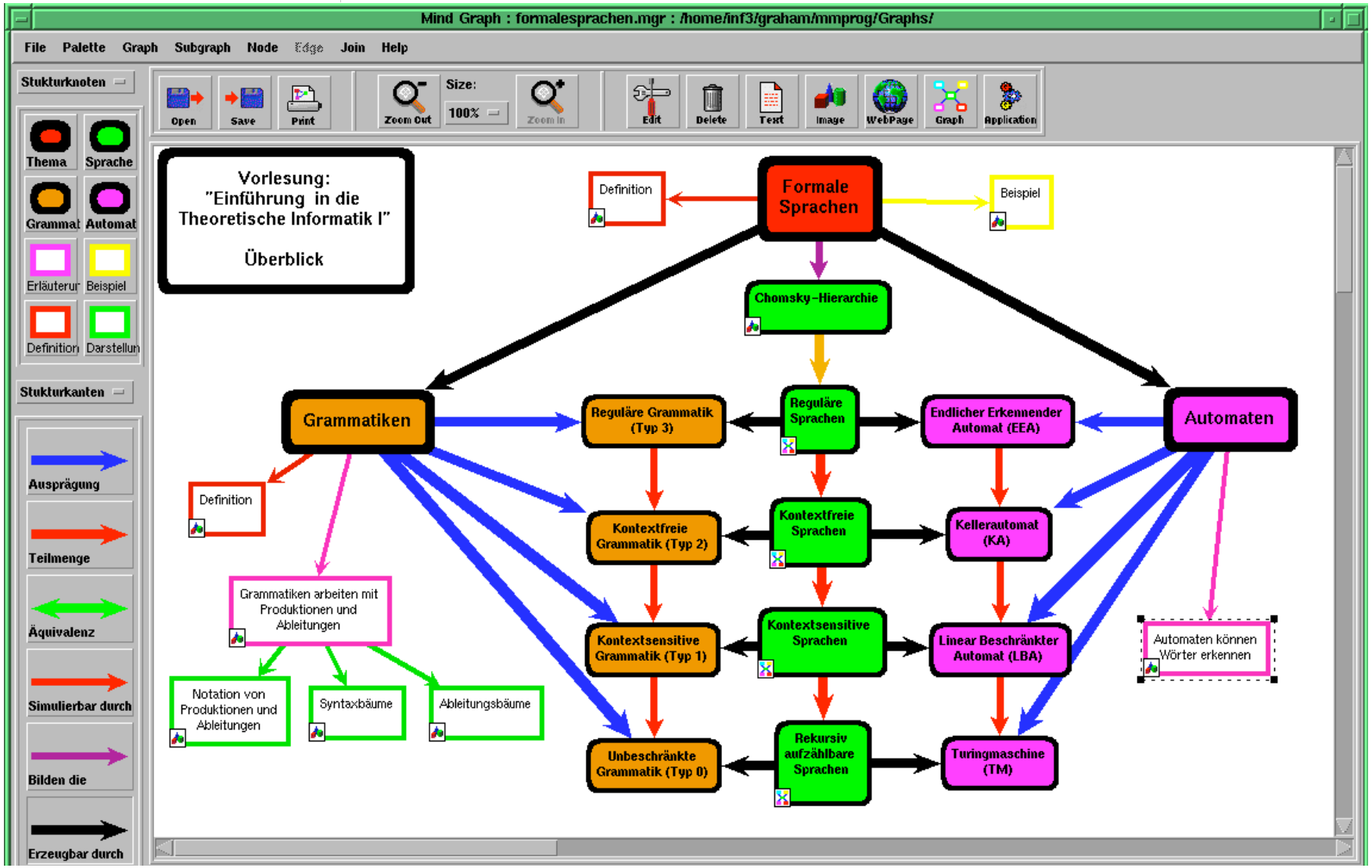
Formality for Modeling Languages

Informal language, minimal syntax, no primitive terms or semantics	e.g., graphs
Informal language, minimal syntax, primitive terms, no semantics	e.g., SADT, DFDs
Formal language with primitive terms, syntax and semantics	e.g., EER, some OOA models
Formal language, with a primitive terms, syntax, semantics and an assertional sub-language for user-defined constraints and rules.	e.g., RML, KAOS

Informal Models

- Diagrammatic notations for sketching an application.
- Offer boxes (nodes) and labelled arrows (links).
- There is no guidance on what boxes and arrows to use for a particular application.
- There is no agreed upon semantics attached to either the boxes or the arrows; users interpret them in terms of their labels, other informal documentation, as well as their own perspective



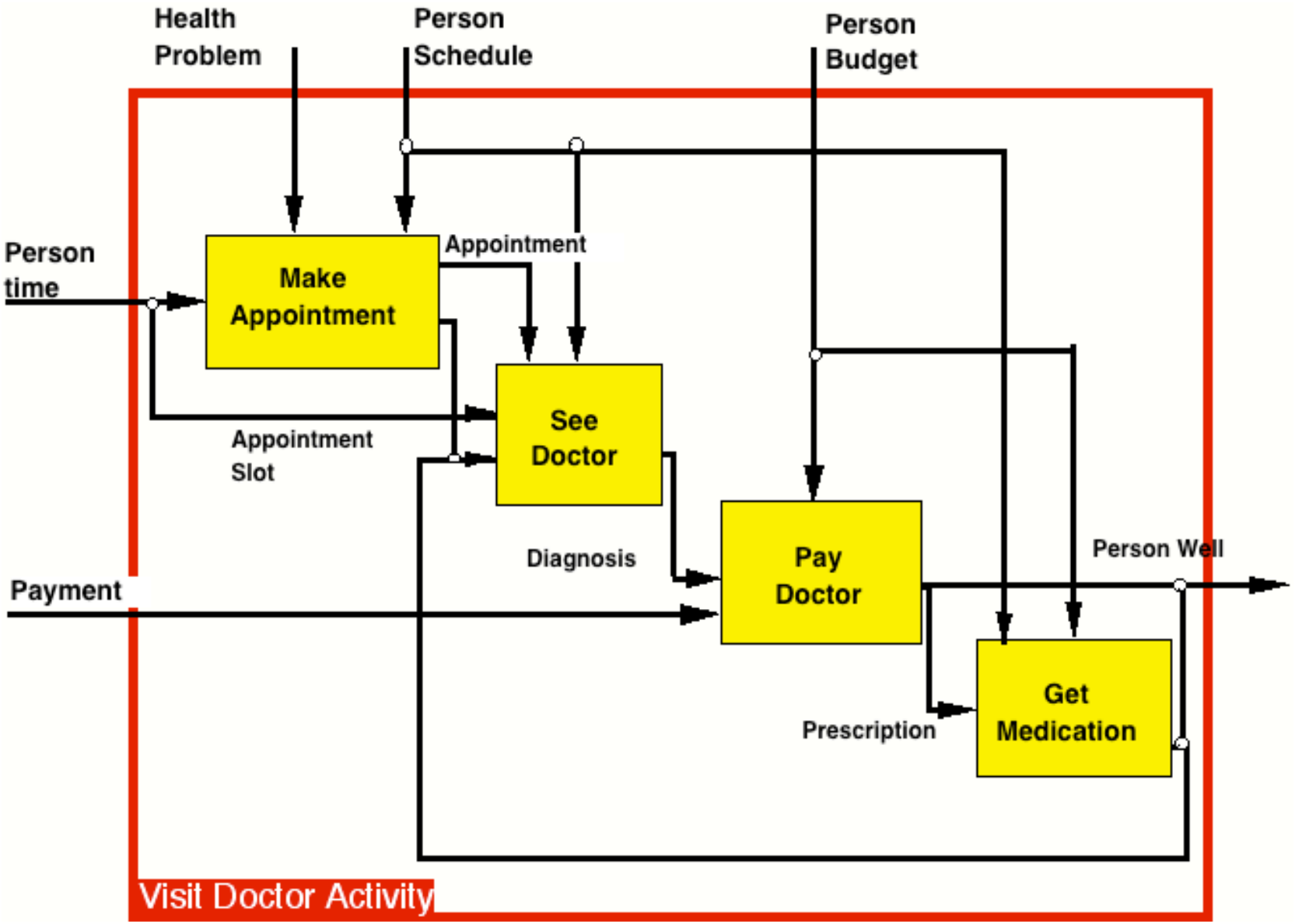

Mind Graph : formalesprachen.mgr : /home/inf3/graham/mmprog/Graphs/
File Palette Graph Subgraph Node Edge Join Help
Open Save Print Zoom Out Size: 100% Zoom In Edit Delete Text Image WebPage Graph Application
Stukturknoten
Thema Sprache
Grammat Automat
Erläuterur Beispiel
Definition Darstellun
Stukturkanten
Ausprägung
Teilmenge
Äquivalenz
Simulierbar durch
Bilden die
Erzeugbar durch
Vorlesung: "Einführung in die Theoretische Informatik I"
Überblick
Definition
Formale Sprachen
Beispiel
Chomsky-Hierarchie
Grammatiken
Reguläre Grammatik (Typ 3)
Reguläre Sprachen
Endlicher Erkennender Automat (EEA)
Automaten
Definition
Kontextfreie Grammatik (Typ 2)
Kontextfreie Sprachen
Kellerautomat (KA)
Grammatiken arbeiten mit Produktionen und Ableitungen
Kontextsensitive Grammatik (Typ 1)
Kontextsensitive Sprachen
Linear Beschränkter Automat (LBA)
Automaten können Wörter erkennen
Notation von Produktionen und Ableitungen
Syntaxbäume
Ableitungsbäume
Unbeschränkte Grammatik (Typ 0)
Rekursiv aufzählbare Sprachen
Turingmaschine (TM)

Diagrammatic Models

- Offer a graphical notations for modeling; they come with primitive terms for modeling an application, i.e., a set of concepts in terms of which we can conceptualize an application

  e.g., SADT offers *data* and *activity* boxes, *input*, *output*, *control* arrows

- There are only *informal semantics* attached to either the boxes or the arrows; users interpret them in terms of their labels, other informal documentation.
- Such techniques usually come with a methodology on how to use them.


Health Problem
Person Schedule
Person Budget
Person time
Make Appointment
Appointment
See Doctor
Appointment Slot
Diagnosis
Pay Doctor
Person Well
Payment
Get Medication
Prescription
Visit Doctor Activity

Formal Models

→ Used as far back as semantic networks [Findler78], also in semantic data models, such as the Entity-Relationship model.

→ We could provide a formal semantics for the SADT box and arrow types in terms of a set of axioms (or in some other way)

∀ A/Activity, E/Entity, t, t'/TimeInterval

[InputOf(E, A) ∧ InstanceOf(a, A, t) ⇒

∃ e InstanceOf(e, E, t') ∧ Overlaps(t', t) ∧ InputOf(e, a)]

where Overlaps(t', t) means 

→ Such an axiom might be enforced by an SADT++ tool which instantiates activities and data

Can't specify user-defined constraints, yet!

References

- [Borgida89] Borgida, A., Brachman, R., McGuiness, D., and Resnick, L., "CLASSIC/DB: A Structural Data Model for Objects", Proceedings ACM SIGMOD International Conference on the Management of Data, Portland, June 1989.
- [Dardenne93] Dardenne, A., van Lamsweerde, A. and Fickas, S.,"Goal-Directed Requirements Acquisition", in *The Science of Computer Programming 20*, 1993.
- [EER92] Parent, C., and Spaccapietra, S., "ERC+: An Object-Based Entity-Relationship Approach", in [Loucopoulos92].
- [Loucopoulos92] Loucopoulos, P. and Zicari, R., (eds.) *Conceptual Modeling, Databases and CASE: An Integrated View of Information System Development*, Wiley, 1992.
- [Mylopoulos92] Mylopoulos, J., "Conceptual Modeling and Telos", in [Loucopoulos92].
- [Smith86] Smith, B., "The Correspondence Continuum", Proceedings Canadian Society for the Computational Studies of Intelligence (CSCSI) National Conference, Montreal, 1986.