COMPOSITIONAL APPROACH TO PROGRAM FORMALIZATION AND VERIFICATION
(methodological introduction)

Mykola (Nikolaj) S. Nikitchenko
Taras Shevchenko National University of Kyiv

Contents

- ***Introduction***
- ***Methodological aspect of integrative approach***
- ***Basic notions of programming***
- ***Formalization of programming notions***
- ***Integrating programming with computability theory***
- ***Integrating programming with mathematical logic***
- ***Conclusions***

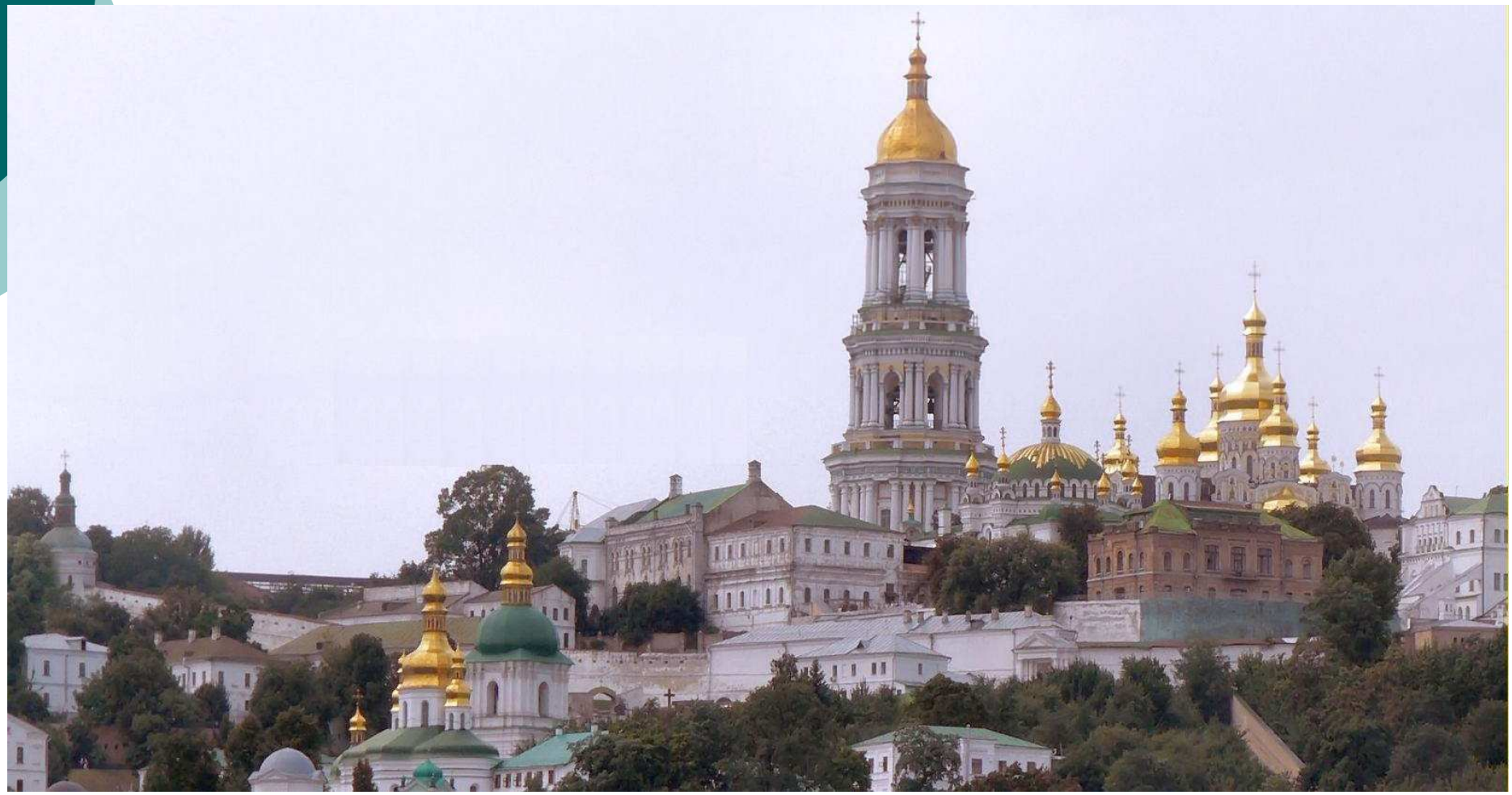
Taras Shevchenko National University of Kyiv

Southern campus of the university

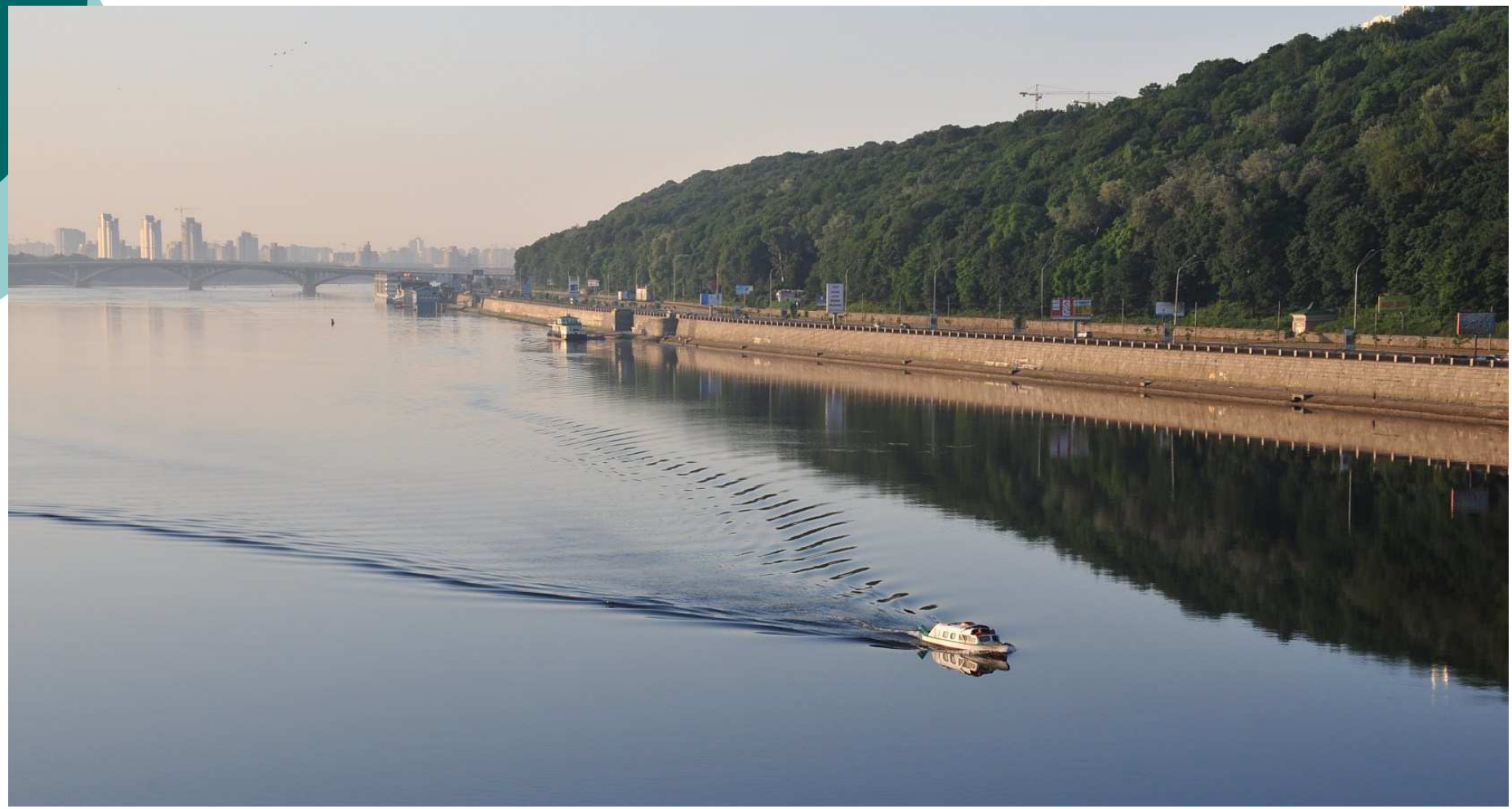
Faculty of Cybernetics

View on Maydan Nezalezhnosti

View on Kiev-Pechersk Lavra Monastery



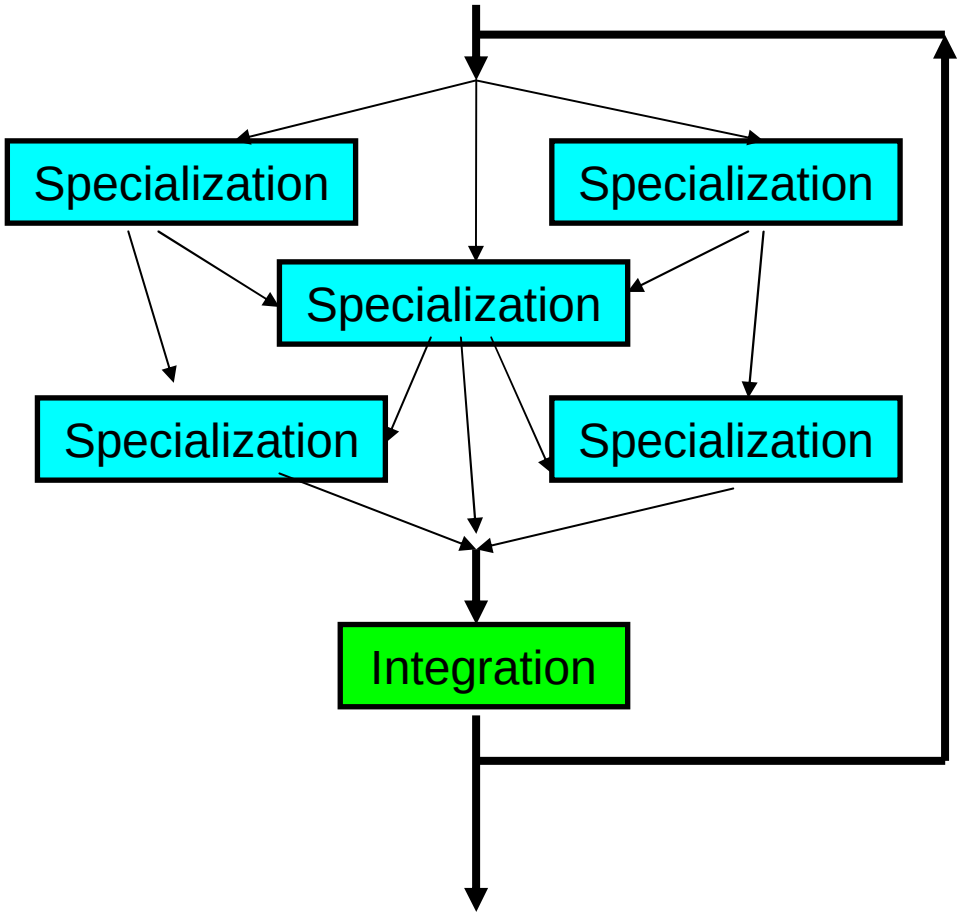
View on river Dnieper



Introduction

- ***In the current computing curricula specialization prevails over integration***
- ***This leads to some negative consequences***
- ***Specialization and integration should be balanced***
- ***The aim of the lecture is to present an integrative composition-nominative approach to programming-related disciplines***

Specialization-Integration Cycle in Theories Development



Integration between Formal Methods

Prof. Wolfgang Schreiner*:

The RISC ProgramExplorer was developed to provide a close integration between programs, theories, specifications, and semantic models.

This is "horizontal" integration.

Next step - "vertical" integration

*Computer-Assisted Program Reasoning Based on a Relational Semantics of Programs

Goals of Integrative Approach

- ***Scientific:*** ***Explication and formalization of semantic-based methods of software system development***
- ***Educational:*** ***development of a new content for computer science disciplines "around" programming***
- ***Practical:*** ***Construction of software and educational systems based on the proposed integrative approach***

Integrative approach (educational aspects)

Aim: construct main parts of programming-related disciplines in integrity of their essential aspects using a relatively small number of

- ***methodological principles,***
- ***basic notions, and***
- ***formal models.***

Integration strongly correlates with fundamentalization that emphases importance of fundamental, basic notions for professional education

Programming-related disciplines

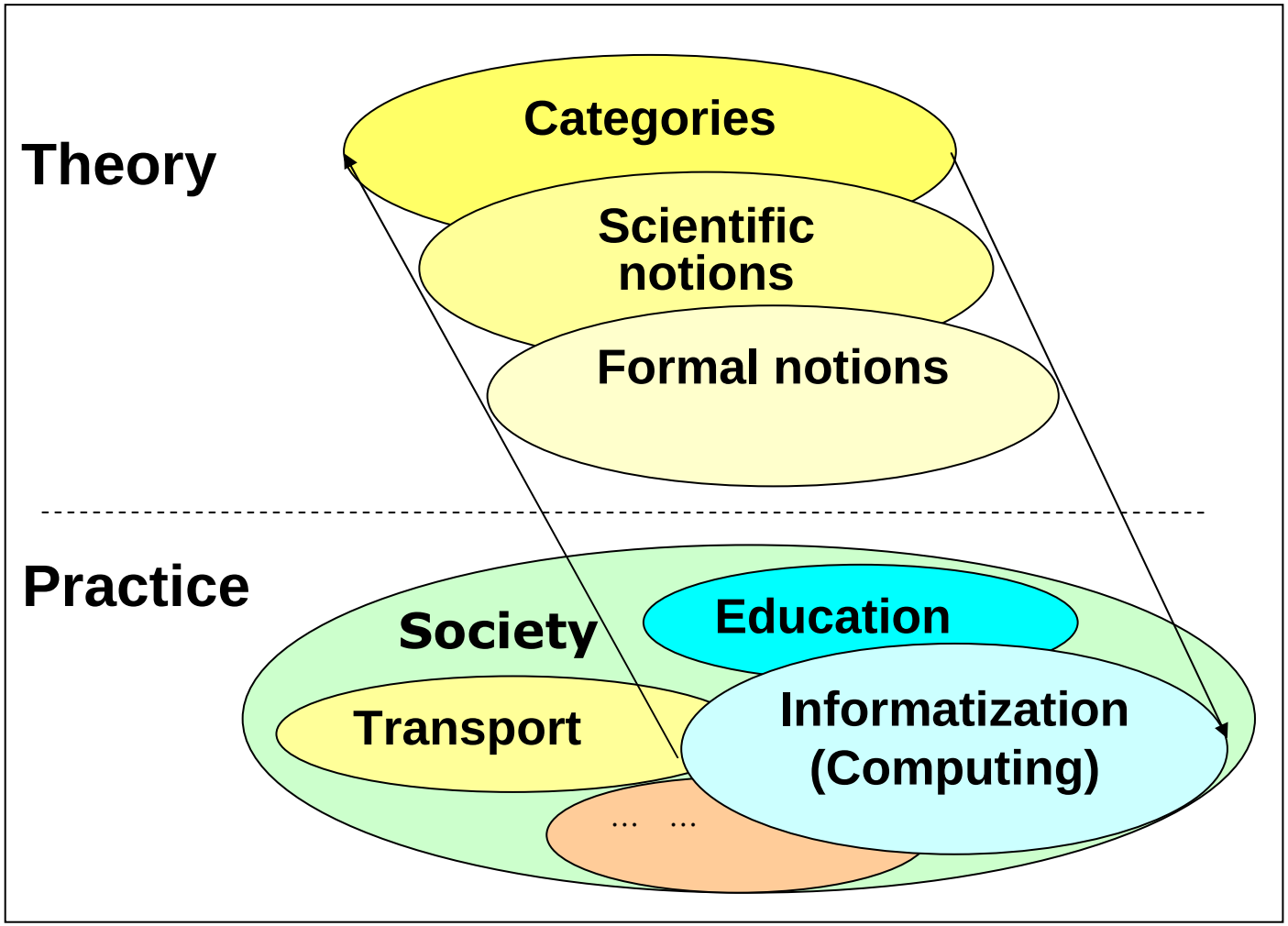
They include disciplines of three groups:

1) ***concerning programming itself,***
2) ***basic for programming like theory of algorithms (computability theory), mathematical logic, universal algebra, theoretical linguistics, and***
3) ***based on or involving programming like system specification, validation and verification, formal methods of software development, requirement analysis, etc.***

Methodological principles

- ***Principle of universal connection*: everything is connected with something else.**
- ***Principle of development from abstract to concrete* (from simple to complex, from a lower level to a higher one, from the old to the new).**
- ***Triadic principle of development*:**
  thesis – antithesis – synthesis
- ***Principle of unity of theory and practice* (variant: *union of logical and historical development*).**

Integration of theory and practice in notion explication



Summary of the proposed approach

- Integration
- By Development
- From Abstract to Concrete
- From Methodological via
  Professional to
  Mathematical Level
  (vertical integrity)
- With Internal Integrity on each Level
  (horizontal integrity)

Main Subject-Object Relations (philosophical level)

- ***Ontological***
- ***Gnosiological (Epistemological)***
- ***Praxeological***
- ***Axiological***
- ***Phenomenological***
- ***…***

We advocate importance of teaching philosophy (in view of knowledge-based economy)

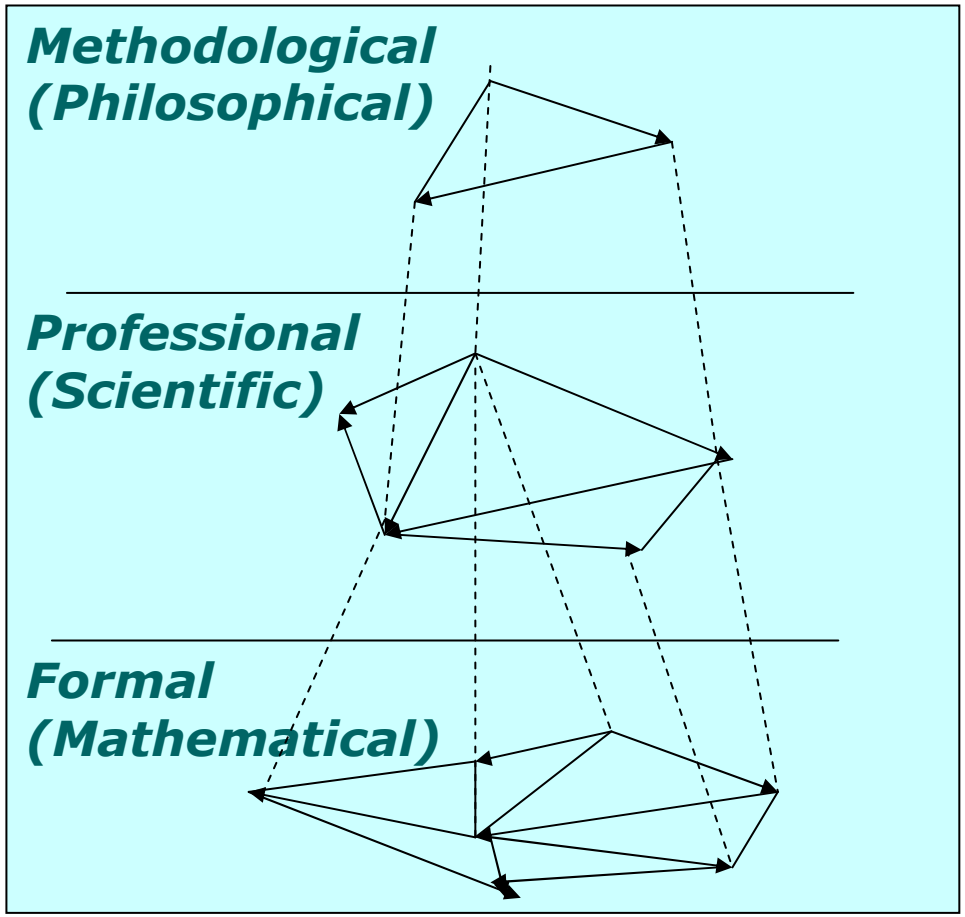
Expected Results (ontological level)

- **Net of Notions (Ontology)**
- **on various levels**
- **with relations between them**

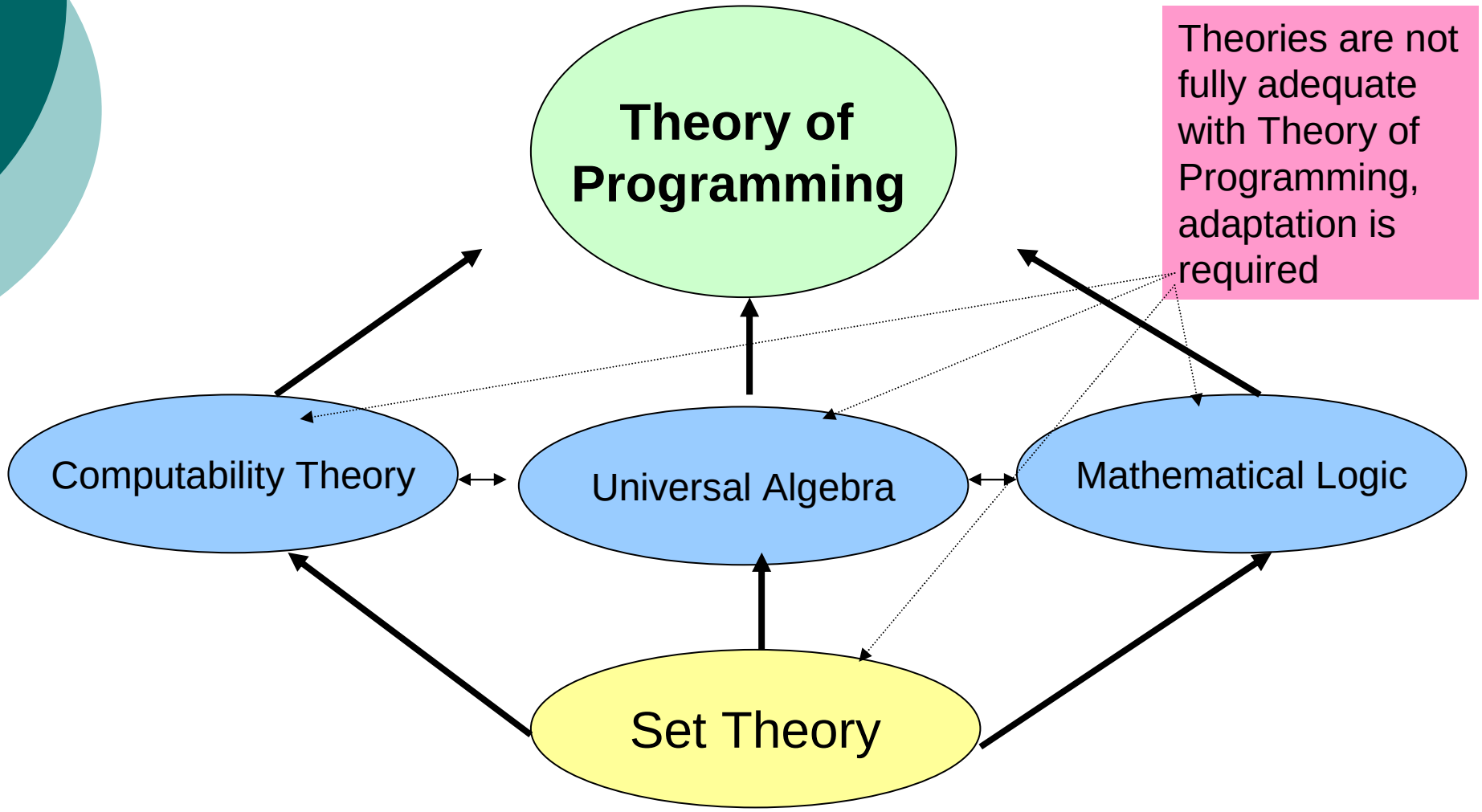
Transformations between levels:

- particularization,

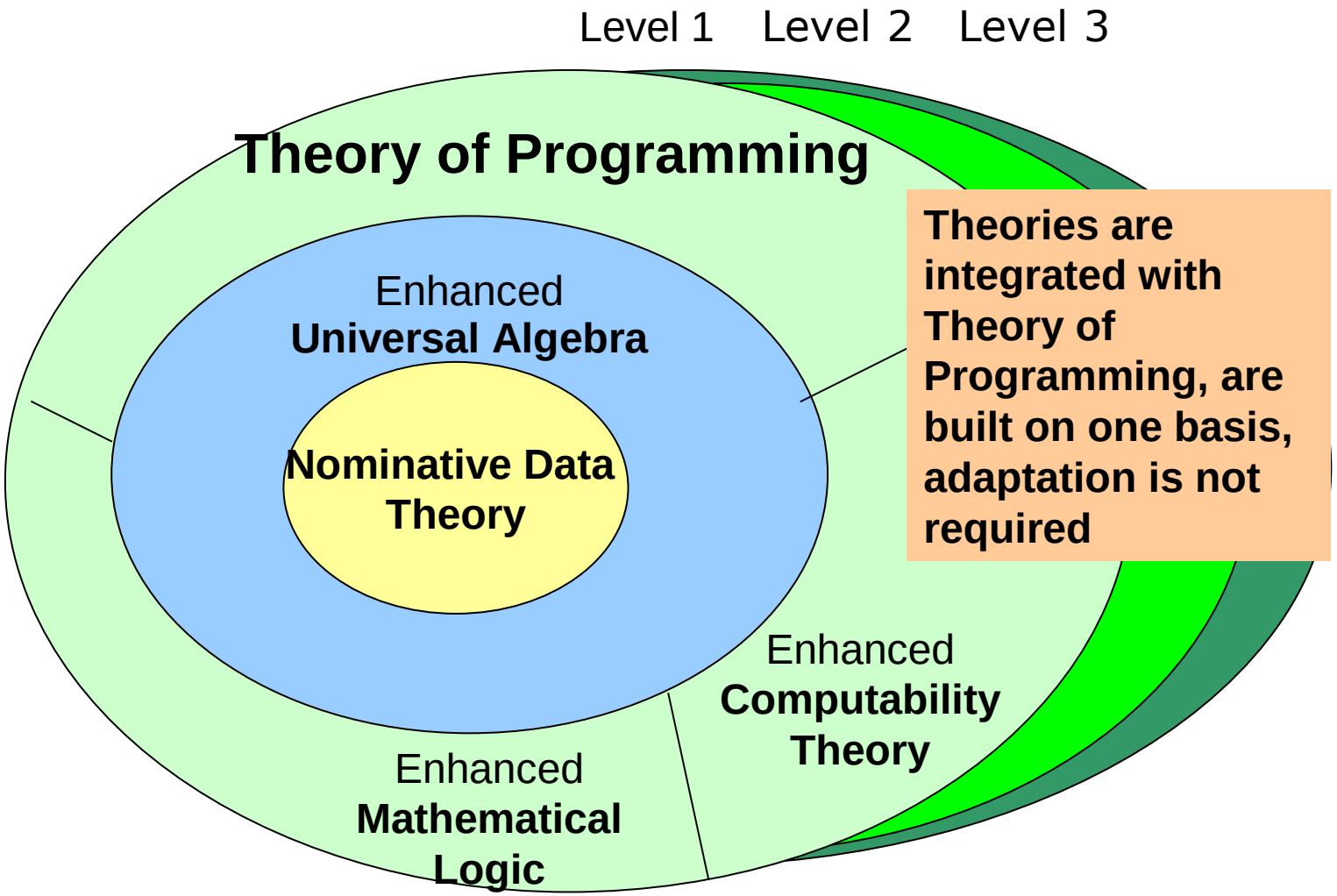
- formalization.



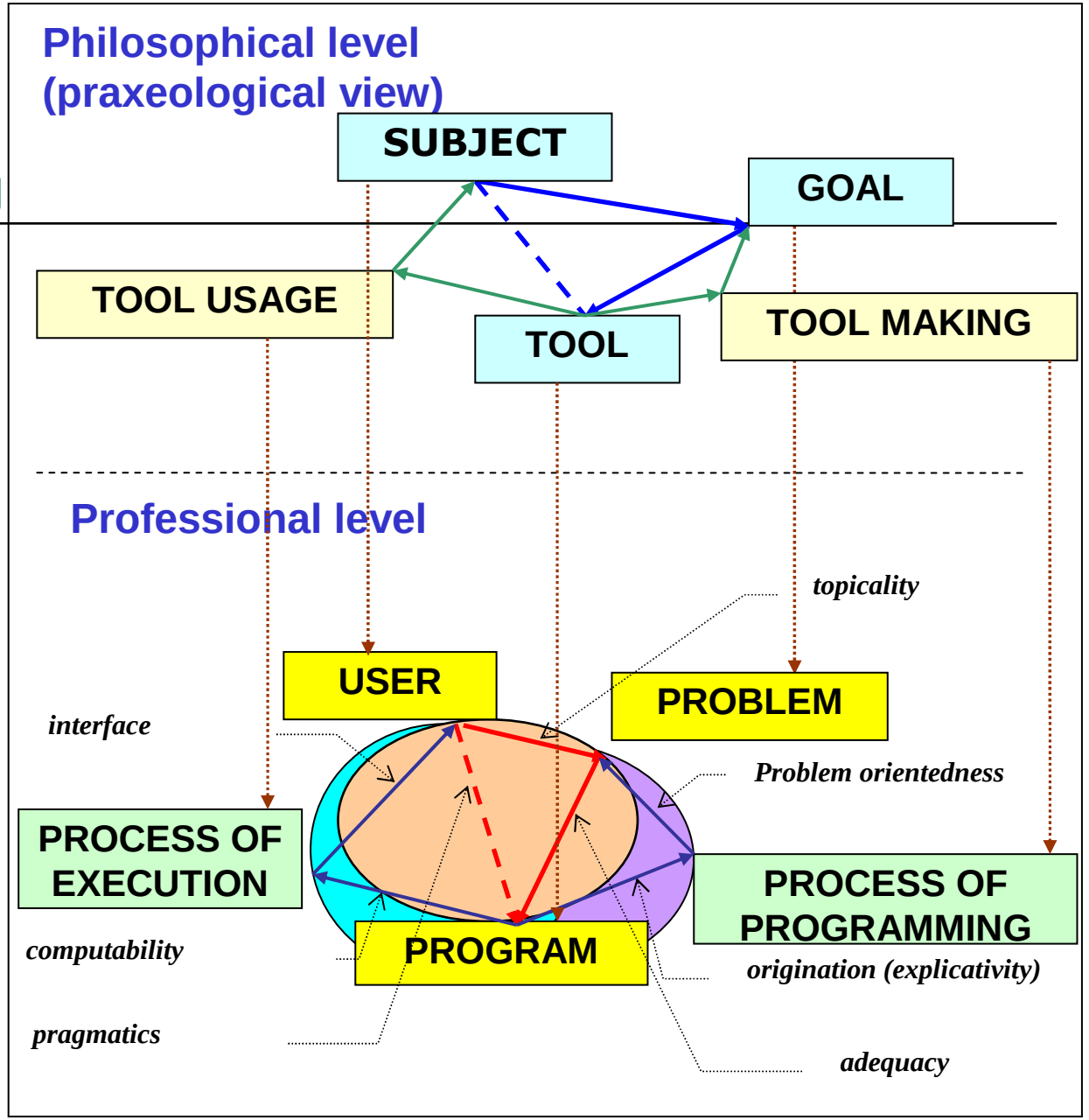
Basic Disciplines for Theory of Programming (mathematical view)



Proposed Dependency Scheme (Algebraic approach)



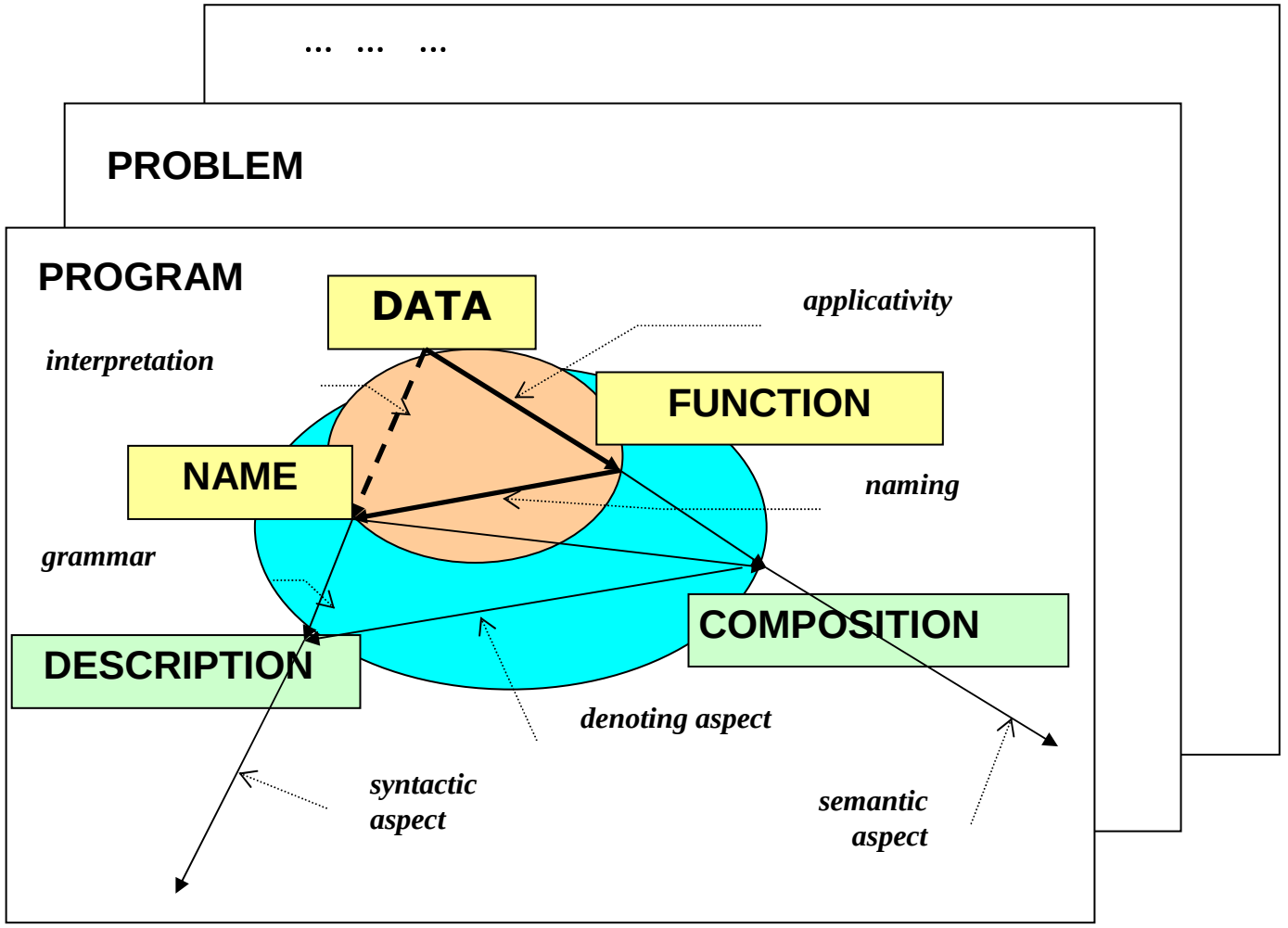
Developing the main notions of Programming



Main Methodological Principles (professional level)

- ***Principle of integrity of intensional and extensional aspects (particularization of categories universal-particular-singular); leading role of intensional aspects***
- ***Descriptivity principle: objects are presented by their descriptions; semantic and syntactic aspects are particularization of categories content-form; leading role of semantics over syntax***
- ***Compositionality principle***
- ***Nominativity principle***

Pentad of the main basic program notions

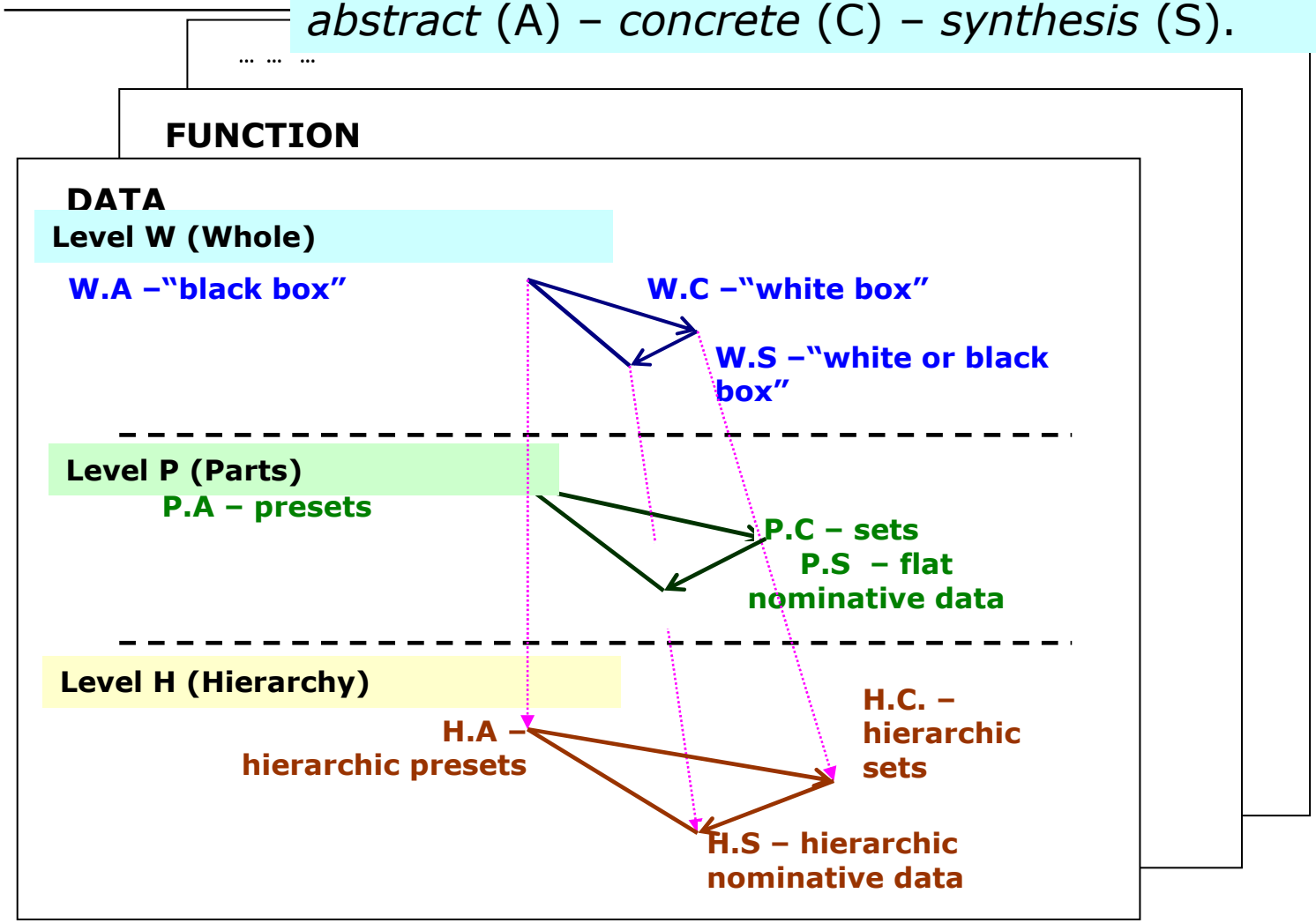


Main thesis (professional level)

The main notion of computer science (informatics) is the notion of *language* (primarily in constructive, formal, communicative, and practical aspects)

Development of the notion of data

Triads of categories:
whole (W) – *parts* (P) – *synthesis* (H as Hierarchy)
abstract (A) – *concrete* (C) – *synthesis* (S).



Nominative data

- **Nominative data are based on the naming relation *name→value***
- **Values can be *simple* (unstructured) or *complex* (structured)**
- **Names can be *simple* or *complex***
- **Names and values can be *independent* (direct naming) or *dependent* (indirect naming is allowed)**

Representation principles

Data representation principle:

program data can be represented as concretizations of nominative data.

Semantics representation principle:

program semantics can be represented by functions over nominative data (nominative functions) constructed with the help of compositions

Formal language model (mathematical level)

The first formal language model –

Composition-Nominative Model:

- ***Semantic (Composition) System***
- ***Syntactical System***
- ***Denotational System***

Composition System:

Data – Function – Composition

Intensions should be taken into account

Semiotic Aspects of Programs

- ***pragmatic***
- ***semantics***
- ***Syntax***

Semiotic aspects are too abstract, pragmatics is overloaded with various senses.

Richer theory of aspects is required

Essential Program Aspects

- ***External program aspects: adequacy, pragmatics, computability, and origination;***
- ***Internal aspects: semantics, syntax, and denoting relation***
- ***Relations between external and internal aspects (process of programming and composition, process execution and function application, etc.)***

Integrating programming with computability theory

- ***Traditional computability is understood as computability of n-ary functions defined on integers or strings (Turing computability, fixed intension).***

- ***The notion of computability over classes of data with different intensions is required.***

Natural computability

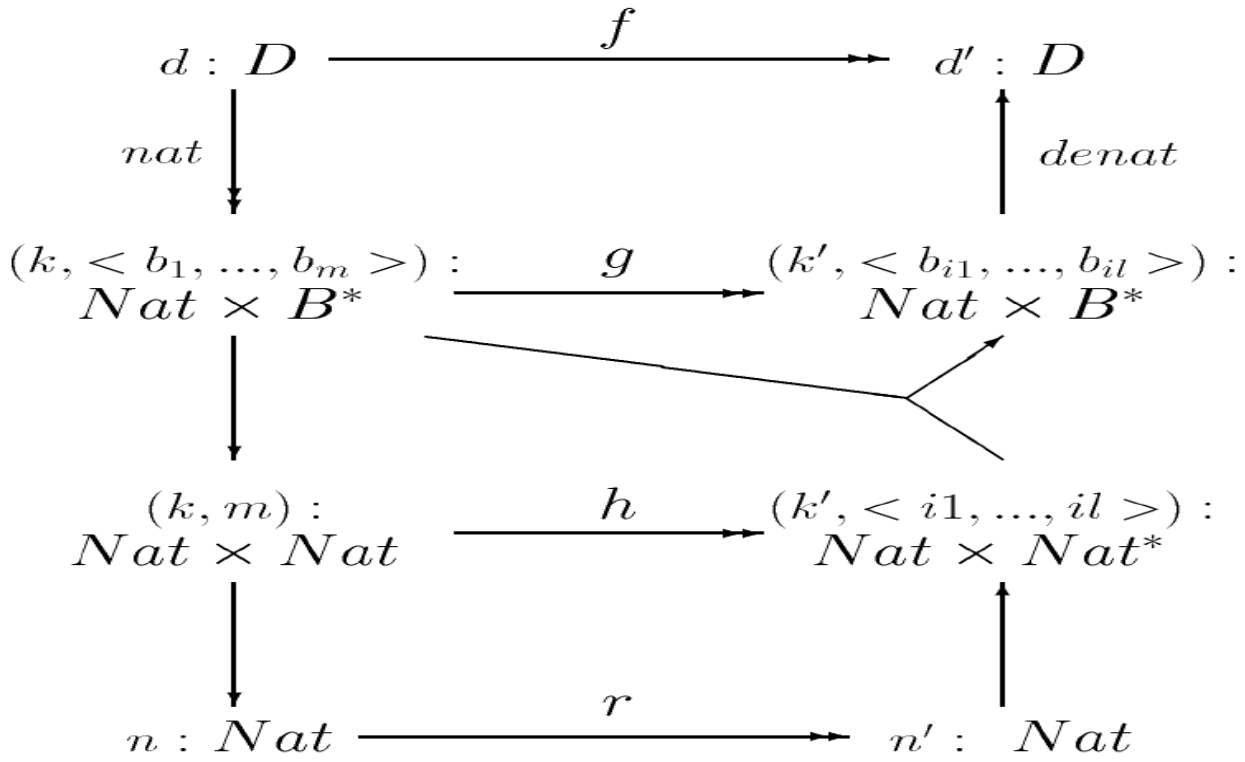


Diagram of natural computability

Complete classes

Theorem 1. $Comp(IA, D)=\{\perp, id\}$.

Theorem 2. $Comp(IC, D)=D\rightarrow D$.

Theorem 3. $Comp(IAC, A \cup C)=\{f \cup g \mid f\in(\{\perp_A, id_A\}\cup\{ A \mid c \in C \}), g\in\ C \rightarrow C \}$.

Theorem 4. $Comp(IND, ND(V,W))= =CLOS(\{\Rightarrow v_0,\ldots, \Rightarrow v_m, v_0\Rightarrow,\ldots, v_m\Rightarrow, v_0!,\ldots, v_m!\}, \{\circ, *, \nabla\})$.

Theorems describes executable components of program specification

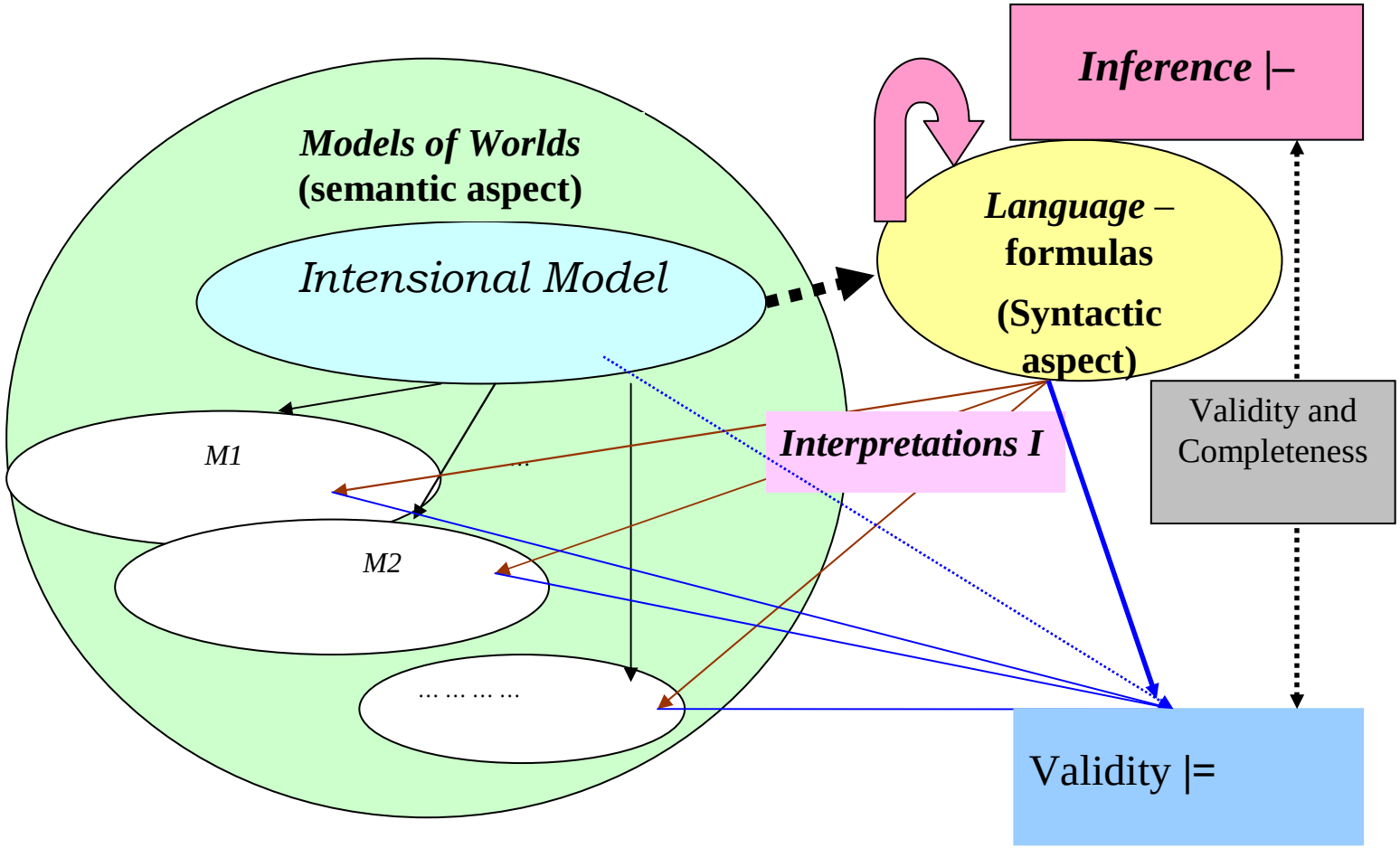
Integrating programming with mathematical logic

The main notions of logic:

(*IMW, ECW, L, Int,* |=, |–)

- *IMW is an intensional model (of worlds),*
- *ECW is a class of extensional models,*
- *L is a language of a logic,*
- |= *is a validity relation, and*
- |– *is an inference relation.*

Main notions of Logic



Classes of logics

With respect to the intensions of data we can specify the following predicate logics:

- *propositional logics (abstract data),*
- *singular logics (concrete data),*
- *renominative, and*
- *quantified logics (nominative data).*

For all these logics composition-nominative languages of predicates are defined and investigated.

Next level –program logics.

Conclusions

Programming-related disciplines:

- ***Theory of Programming***
- ***Theory of Algorithms***
- ***Mathematical Logic***
- ***Universal Algebra***
- ***Specification and Programming Languages***
- ***Formal Methods of Software Development***
- ***Databases***

can be taught on *one (methodological, professional and formal) basis.*

Constructed models are formal thus permitting their thorough investigation with further implementation of e-learning tools.

References

Formal definitions are presented in:

- *Nikitchenko N.S.* A Composition-Nominative Approach to Program Semantics.– IT-TR: 1998-020.– Technical University of Denmark.– 1998.– 103 p.
- *Nikitchenko M., Shkylniak S.:* Mathematical logic and theory of algorithms: Handbook. Publishing house of National Taras Shevchenko University of Kyiv, 528 p. (2008) (in Ukrainian).
- *Red'ko V., Brona J., Buy D., Poliakov S.* Relational Databases. – K. Akademperiodika, 2001.– 198 c. (In Russian)
- *Basarab I., Nikitchenko N., Red'ko V.* Composition Databases. –K.: Lybed', 1992. – 191 p. (In Russian)

Thank you!
Questions?