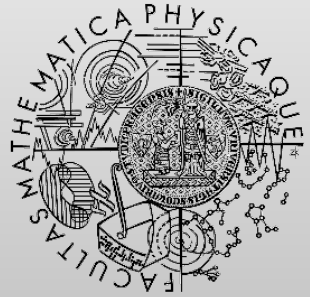
Prototype-based languages

http://d3s.mff.cuni.cz



FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Tomas Bures

bures@d3s.mff.cuni.cz

IO language

IO

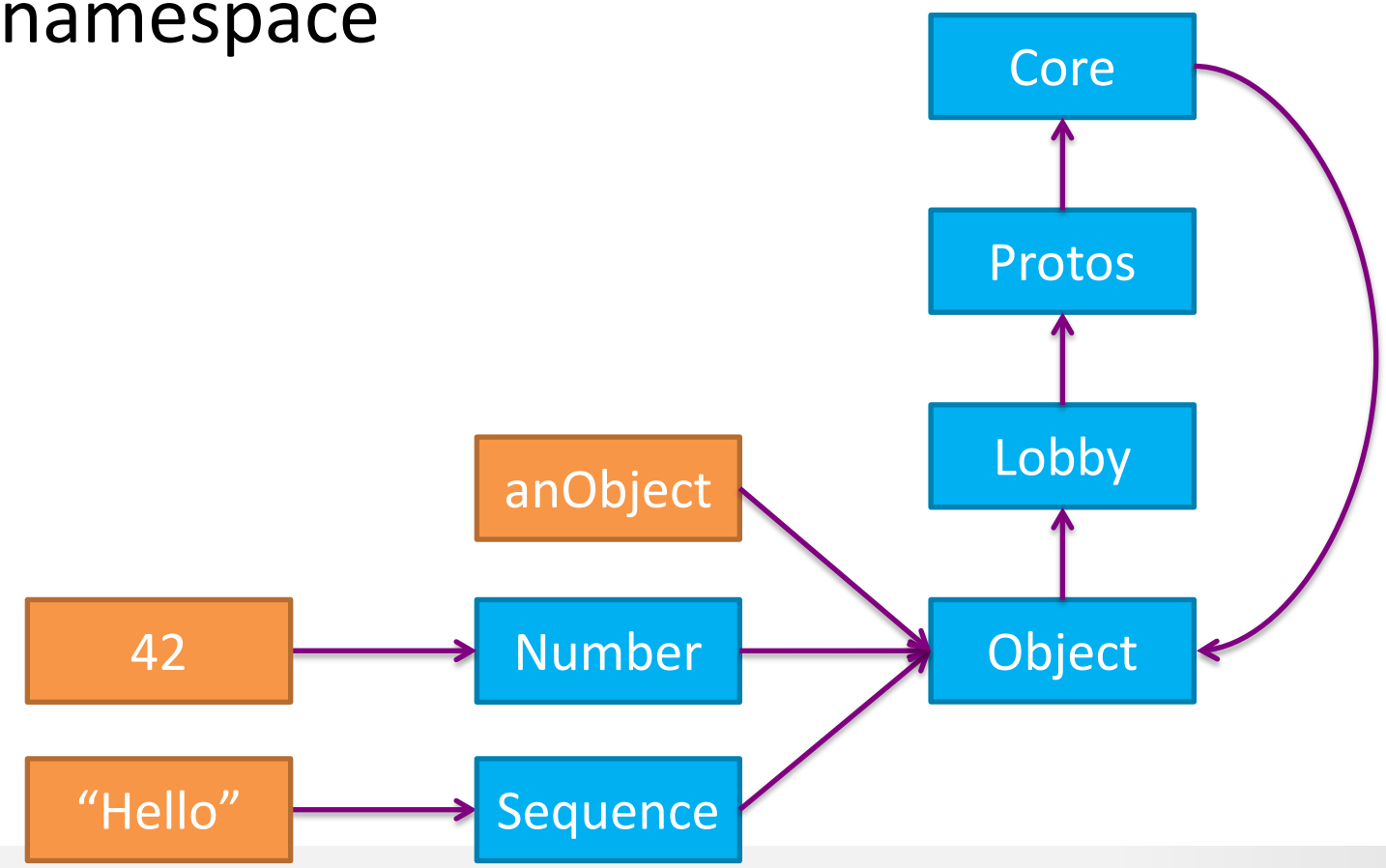
- Dynamic prototype-based programming language
  - All values are objects
  - No classes
  - Differential inheritance
  - Code is a runtime inspectable / modifiable tree
    - Essentially a list of messages

Basic concepts

- An object is a set of slots
- Object responds to messages
  - Messages handled by anonymous function stored in a slot with the name of the message
  - Properties are accessed via messages getSlot, setSlot, and updateSlot
  - :=, = are short-hand forms of updateSlot and setSlot
- Example: io01

Basic concepts

- Each object has a list of prototypes
  - Consulted in depth-first search order when a lookup in the object table fails
- Lobby is the global namespace for objects
- Example: io02



Basic concepts

- New objects created by cloning
  - Cloning creates a new object and sets the proto link to the object being cloned
- Differential inheritance
  - An object contains only attributes which are different to its prototype
- Slots can be added to any object
- Example: io03

Messages

- Code is composed of a sequence of messages
  - Each message has a name and list of arguments
  - Each argument is again a message
- Message is evaluated in a context of an object
- Example: io04

Methods / Blocks

- A block/method is a message with associated scope and parameters
- Return value is the last message in a sequence
- When invoked, activation record is created with
  - Actual parameters
  - ‘call’ object
    - ‘call target’ – target object of the call
    - ‘call sender’ – sender object
    - ‘call message’ – message used to invoke the call
  - ‘self’ – reference to the scope
  - Forward to ‘self’ for all failed lookups
- Example: io05

Methods / Blocks

- Method
  - Activatable block – called when accessed
    - Accessing without calling via getSlot(name)
  - With scope := nil – scope is set to the target object
- Block
  - Not activatable by default
  - Scope set to target of the ‘block’ message
  - Serve as local scopes within the lexical scope
- Example: io06, io07

Methods / Blocks

- Invoking a block/method means evaluating its message in a given context

- Example: io08

Control structures

- Control structures (if, while, for, ...) are ordinary methods
  - Can be implemented in the language
  - Thanks to message abstraction of the code
  - In fact ‘method’ is also an ordinary method
- IO thus has very minimal syntax and no keywords

- Example: io09