IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Introduction to languages & logic

ELU 610 – C1

1^{st} semester 2019

An introduction to…

- mathematical tools for computer science
- two new programming paradigms
- compilation and typing
- tools for knowledge representation

0. Introduction (C1, now)
1. Regular expressions, automata, formal grammars
   - C2-4, TP1-3
   - Éric Cousin – office D03-014, `eric.cousin@imt-atlantique.fr`
2. λ-calculus, functional programming, compilation, typing, OCaml
   - C5-7, TD1-2, TP4-9
   - Fabien Dagnat – office D03-120
   - Jean-Christophe Bach – office D03-124

     `{fabien.dagnat,jc.bach}@imt-atlantique.fr`
3. Logics
   - C8-11, TD3-5, TP10-11
   - Yannis Haralambous – office D03-118,

     `yannis.haralambous@imt-atlantique.fr`

Evaluation

- Theory
  - each part is evaluated at the end of ELU610, june 6th
- Practice
  - a stack language compiler written in OCaml
  - 2 members per compiler group
  - not handing (or contributing to) the compiler is eliminatory

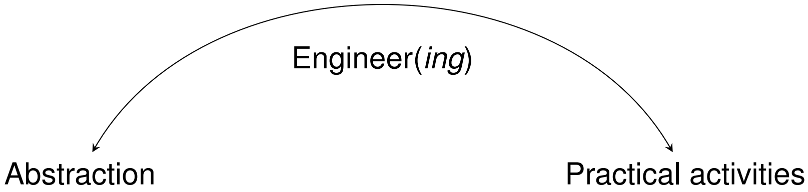
Why *Languages & logic*?

- what relationship between language and logic?
- why that content in a lecture?
- why studying formal systems and abstractions instead of practical activities?

Today's lecture: motivations for L&L

+ few formal definitions
+ some terminology

Abstractions *vs* practical activities



- Playing with (changes of) abstractions is part of engineer's core activity
- Reasonable pedagogical choice

- Complex software systems
- Critical systems
- Need of trusted software for trusted systems and services

$\Rightarrow$ designing, developing and verifying software

{quality, safety, security} by design

$\Rightarrow$ need of tools and methodologies

A problem well stated is on its way to solution

Bergson, XXth

- What does *stated* means?
- What is a *well* stated problem?
- Then *solving* it…

- What tools do we have to state problems?
  - natural language, pictures/drawings, mathematics, programs
  - media (audio – voice, paper – writings, drawings, electronics. . .)
- How do we state problems?
  - identification
  - selection
  - description (with a sound, a word, a picture, a formula, a program, etc.)

- Recognize, invent
  - with respect to previous identical experiences
  - with respect to previous close experiences
- Similarities, metaphors, links, comparisons
- Exact, approximate, complement (lack of), ...
- An invention from scratch is rare, ... (is it even possible?)

In the beginning was the Word

John the Apostle, Ist (?)

Mal nommer les choses, c'est ajouter au malheur du monde

Albert Camus

- The importance of choosing right names
  - ambiguities, vagueness
  - method overriding/overloading

- Among the identified *things*, which one to keep?
  - all?
  - the useful one? Useful relatively to an *intent* (the problem to solve)
- Ockham's razor

  Entities must not be multiplied beyond necessity.
  Plurality should not be posited without necessity.

  William of Ockham, XIVth

An usual interpretation is: "when you have two competing theories that make exactly the same predictions, the simpler one is the better"

To describe an idea in order to:

- Transmit (in time, to others, to oneself)
- Handle, work with
- To interpret, with risks like:
  - incomprehension… easy; "I'm able to detect when I do not understand!"
    - lost languages or writings
  - misunderstanding… "I understood something, but not the intent of the transmitter" hard to detect… but a factor of innovation

Our ability to identify, to select and to name depends on our toolbox of descriptions

(Scientific) progress is a consequence of this virtuous principle:

> *We are like dwarfs on the shoulders of giants, so that we can see more than they, and things at a greater distance, not by virtue of any sharpness of sight on our part, or any physical distinction, but because we are carried high and raised up by their giant size*

Bernard de Chartres, XIIth

Describing how?

- Sounds
  - problems: trace, memory, transmission, sophisms (validity, correctness)
- Writings
  - problems: sophisms (validity, correctness)
- Graphical
  - problems: validity (interpretation/semantics)
- Mathematics
  - problems: accessibility, calculability/completeness
- Computers
  - problems: validity – 4-colors theorem (?), size of problems, calculability/completeness

- Modeling
  - abstracting a problem, stating it...
  - simplifying, hiding details
- What for?
  - solving problems (of course!)
  - helping to think
  - mastering complexity
  - validating
  - verifying
- How to...
  - ...express a model / represent concepts?
  - ...how to “solve” a problem with models? (how to reason?)

- Modeling
  - abstracting a problem, stating it…
  - simplifying, hiding details
- What for?
  - solving problems (of course!)
  - helping to think
  - mastering complexity
  - validating
  - verifying
- How to…
  - …express a model / represent concepts?

  $\Rightarrow$ with languages
  - …how to “solve” a problem with models? (how to reason?)

  $\Rightarrow$ with logics

- Mathematics
  - rich, precise, rigorous
  - possess powerful transformation tools
  - ex.: from $5 + x = 8$ one reduce $x = 8 - 5$, hence $x = 3$!
- Maps, pictures
  - rich, abstract
  - "One picture is worth ten thousand words"
  - transformations (3D algorithms, drawing constraints)
- Simulations
  - models (french *maquettes*), prototypes
  - actors, virtual reality, ...
- ...

- Modeling
  - choose a good language (to be able to express concepts)
  - symbols, graphical notations
  - mechanisms, operations
- Example in math, using “algebra” (no verb!?)
  A square has a surface a. What is the length of its side?
  - x is the length of the side
  - x is such that $x^2 = a$
  - …do not forget $x \geqslant 0$!
- Defining a language needs time

Example : math notations in history [Nic94]

Authors	$+$	$=$	x	$2x^2 = 3x + 5$
Chuquet (XVth)	$\bar{p}$		$1, 2, 3$	2^2 egaulx a 3^1 $\bar{p}$ 5
Stifel (XVIth)	$+$		x, z, a	$2z$ acquatus $3x + 5$
Cardan (XVIth)	$\bar{p}$		*co*, *ce*, *cu*	2 *ce* equale a 3 *co* $\bar{p}$ 5
Bombelli (XVIth)	$\bar{p}$		$\underline{1}, \underline{2}, \underline{3}$	$\frac{2}{2}$ equale a $\frac{1}{3}$ $\bar{p}$ 5
Stevin (XVIth)	$+$		$\textcircled{1}, \textcircled{2}, \textcircled{3}$	$2\textcircled{2}$ aequatus $3\textcircled{1} + 5$
Viète (late XVIth)	$+$		*A*, *Aq*, *Ac*	2 in *Aq* aequatur 3 in *A* $+$ 5 plano
Neper (XVIIth)	$+$	$==$	R, Q, C	$2Q == 3R + 5$
Harriot (1631)	$+$	$==$	a, aa, aaa	$2aa == 3a + 5p$
Hérigone (1634)	$+$	$2/2$	$a, a2, a3$	$2a2\ 2/2\ 3a + 5p$
Descartes (1637)	$+$	∞	z, zz, z^3	$2zz \infty 3z + 5$

Good modeling language? Criteria

- No ambiguities (2/2, 1, 2, etc.)
- Generalizable (1 to n unknowns)
- Simple (5 plano is redundant)
- Economical (short)
- Ease communication/easy to learn

Cognitive gap: naming what is known is natural; naming the unknown, less. . . !

Everything is about language

- to express
- to reason about

Which language to use?

- universal language? $\Rightarrow$ universal tool
- specialized languages? $\Rightarrow$ dedicated tools
- natural languages? $\Rightarrow$ tools?

Why not using natural (not formal) language?

- ambiguities
- under-specification (understatement, implicit)
- over-specification (redundancy)
- noise
- easy to have contradiction
- difficult to have the right level of specification

$\Rightarrow$ difficult to reason with natural languages

DSL: Domain Specific Language

- special purpose languages...
- on purpose language limitations (Controlled Natural Language)

$\Rightarrow$ Specialized tools for reasoning, transforming, proving, ...

- removing/avoiding ambiguities
- automating reasoning (partially)

$\Rightarrow$ useful for software verification

- formality with 3 levels of correctness:

1. $2x + = 8-$ (syntactic)
2. $2x = 10 \Rightarrow x = 10 - 2$ (transformation)
3. $2x = 10 \Rightarrow x = 10/2 = 5$ (intention)

Levels 1 and 2 can be automated.
Level 3 requires interpretation, and some kind of agreement (consensus); is the problem well stated?

> *[A proof] is a social process that determines whether mathematicians feel confident about a theorem [DLP78]*

Defining a language

- Language = syntax + semantics
- Syntax
  - we have tools to describe syntax without any interpretation:

  $\Rightarrow$ formal grammars
  - writing programs which recognize syntactically correct programs

  $\Rightarrow$ compilers
- Semantics
  - what happens when executing
  - two languages can have the same syntax but different semantics
  - interpretation
  - set of rules, transformations and constraints attached to syntax

Note: reasoning and deduction are a purely syntactical process ($\Rightarrow$ useful for automation...)

- Formal definitions
  - language, alphabet, symbols, terms, . . .
  - focus of CS on *finitely generated* languages
- *formal language theory* (study and classification of languages) [ALSU06, HMU06]
  - focus on how to define languages and (efficiently) recognize terms

see http://en.wikipedia.org/wiki/Formal_language

- . . . and many other interesting language-related things

⇒ Do not miss Éric Cousin's lecture! It is mandatory to understand how we work with languages and compilation in CS

Defining a syntax (another *teaser* of next lecture)

- A syntax? *Two* syntaxes: a *concrete* one and an *abstract* one
- Concrete syntax
  - defined by a grammar, using BNF/EBNF

    see `http://en.wikipedia.org/wiki/Backus-Naur_Form`
  - focus on interaction with the user
    - must be readable, efficient, ...
    - must solve the ambiguities, priorities, associativity, ...

    see `http://www.infoq.com/presentations/Language-Design`
- Abstract syntax
  - the *essential* content of a sentence
  - aimed at being used by any tool manipulating terms
  - defined by a signature (using eventually a BNF/EBNF grammar)
- Understanding syntaxes is necessary to build a compiler

⇒ Do not miss Éric Cousin's next lecture

We talked a lot about syntax, few about semantics.
Where is the semantics?

- in your mind first (we are interpreters)
- in the set of rules, transformations, constraints that we attach to a syntax (ex. $+$ is associative and commutative)
- in mappings we make to a well known world with its own syntax and semantics (ex. mathematics)

- There is mainly three ways of defining the semantics of a term
  1. Axiomatic semantics: some logical assertions states properties of terms
  2. Denotational semantics: each term is mapped to an object of a known space
  3. Operational semantics: how computation behaves (the sequence of states)
- Not the only ones

  see http://en.wikipedia.org/wiki/Semantics_(computer_science)

- Defined by systems of equations describing the effect of each syntactic construction to logical assertions
- Gives a macroscopic vision of the meaning (generally partial)
- Used to study properties of: consistency, completeness, compositionality, ...
- The most well-known, Hoare triple
  - $\{Pre\}T\{Post\}$ means if *Pre* is true before the execution of *T* and *T* terminates then *Post* is true after its execution

see http://en.wikipedia.org/wiki/Hoare_logic

- Defined by a projection in a (known) mathematical space (sets, universal algebra, domain, category, ...)
- Gives an abstract vision of the meaning
- Used to study meta-theory: equivalence of terms, fixed-point theory, ...
- Often given by a projection called an interpretation and denoted $[\![.]\!]$ or $\mathcal{I}(.)$
- Often requires compositionality, the meaning of a term is the composition of the meaning of its subterms

see http://en.wikipedia.org/wiki/Denotational_semantics

- Each term either reduce to another (smaller) term or is a value
- Defined by computation rules (rewriting)
- Can be small-step or big-step
- Gives a microscopic vision of the meaning
- Used to study properties of: termination, non-determinism, . . .
- The one we will use
- More precisely, we will use transitions systems (*a.k.a.* reduction)

see `http://en.wikipedia.org/wiki/Operational_semantics`

⚠ The following slides might be a bit shuffled

Do you know those words? Do you know they meaning?

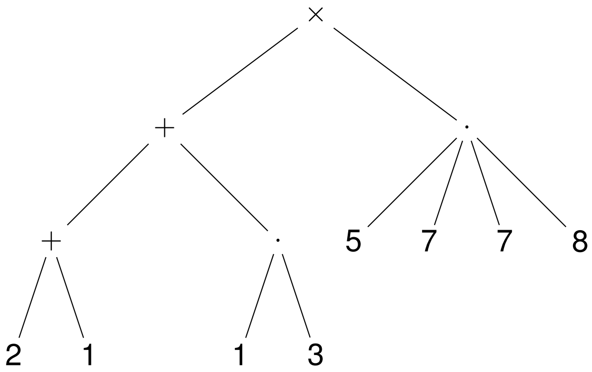
- verification, validation
- term, metaterm
- variable, metavariable
- transition system

An interpretation...

Verification
: checking that the rules of the formal systems are properly used. Internal to a model, a description system and its use.

Validation
: comparing two (2) models to check that the one to be validated *gives the same answer* that the one of reference.

- Terms are trees where each constructor is a node and each of its direct sub-terms is a child



- We use names to
  - represent a set of terms: metavariable; $c(M_1, M_2)$, $E_1 \times (3 + E_2)$
  - represent an unknown part of a term: variable; $c(x)$, $x \times y$
- Metavariables are not part of the syntax and used only inside metaterms
  - a metaterm is a set of term
  - useful to manipulate or to describe properties of sets of terms
- Variables are part of the term
  - syntax must be extended (see next slide)
  - a variable may occur several times within a term
  - the meaning of a variable is given by replacing all its occurrences by a term
  - a term T containing x can be viewed as a function from term to term

- Terms may also contain variables from a denumerable set $\mathcal{X}$
  - we suppose $\Sigma \cap \mathcal{X} = \emptyset$ and the arity of variables is 0
- $T_{\Sigma \cup \mathcal{X}}$ is denoted $T_\Sigma[\mathcal{X}]$
- A term without variable is a ground term (or closed term)
- Variables are leafs (as nullary constructors)

$$\times(+(2, \times(x, y)), x)$$

- The meaning of a variable is given by substitutions

A transitions system is a pair $(S, \rightarrow)$ of a set S (of states) and a binary relation $\rightarrow$ of S ($\rightarrow \subset S \times S$).

A pair (p, q) of $\rightarrow$ is noted infix $p \rightarrow q$ and we speak of a transition from state p to state q.

programs $\leftrightarrow$ transition systems

This introduction could have much more vocabulary. Some will (should) be in the next lectures

- normal term, normal form
- trace, reduction sequence
- (non-)determinism
- strong normalization
- weakly normalizing
- confluence
- reflexive transitive closure
- …

But we are humans… If you hear *strange* (unknown) words which are not defined during the lectures, please tell us.

Conclusion

- an introduction to motivate ELU610
- now, you should understand why there is a lecture combining language and logic
- few intuitions before more formal lectures and definitions
- don't be scary: there are also practical and concrete parts (programming!) to show you that it is useful

Éric Cousin Regular expressions, automata, formal grammars

JC Bach λ-calculus, introduction to functional programming, compilation and typing (OCaml language)

…with Fabien Dagnat

Yannis Haralambous Variation about logics

⚠ Important note: if you do not understand something or if you disagree with us, please say it and ask your questions. We won't bite you, and we follow Crocker's rules[1].

We cannot answer the questions you do not ask…

[1] `http://sl4.org/crocker.html`

References I

Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman.
Compilers: Principles, Techniques, and Tools (2nd Edition).
Addison Wesley, August 2006.

Richard A DeMillo, Richard J Lipton, and Alan J Perlis.
Social Processes and Proofs of Theorems and Programs (Revised Version).
1978.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman.
Introduction to Automata Theory, Languages, and Computation (3rd Edition).
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

Jean-Marie Nicolle.
Histoire des méthodes scientifiques.
Bréal, 1994.