# Lecture V. Reasoning, CGs, and Beyond

# This lecture

1. NL reasoning in proof assistants
2. Dependent Categorial Grammar
   2.1. Introduction to CGs
   2.2. Substructural type theory: introduction (application to syntactical analysis)

# V.1. NL Reasoning in Proof Assistants

- Interactive theorem proving based on MTTs
  - Automatic TP v.s. interactive TP
- An ITP system consists of three parts for:
  (1) contextual defns (2) proof development (3) proof checking

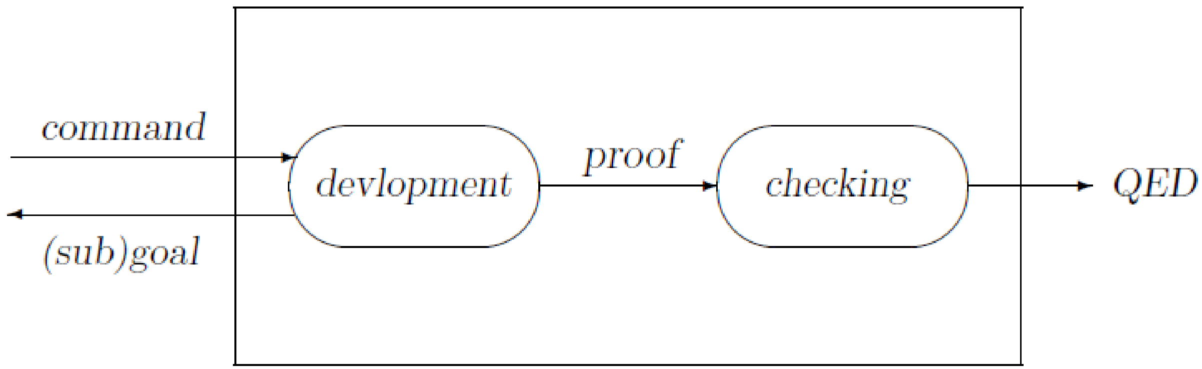



Figure 1: Interactive proof development and proof checking

# Simple example (a theorem about primes)

(* context: properties about primes *)

Definition div (x y : nat) : Prop := exists z : nat, y = x*z.

Definition prime (n : nat) : Prop := n >= 2 /\ (forall x:nat, (div x n) -> x=1 \/ x=n).

(* Theorem: there are infinitely many primes. *)

Theorem inf_many_primes : not (exists n:nat, forall x:nat, prime x -> x < n).

One can then use commands to interact with the system to solve goals by generating “subgoals” and, finally (if successful), to use Qed to finish it.

(Details omitted)

# Proof development process

- Enter: `Theorem tautology : forall (A : Prop), A->A.`

```
1 subgoal
____________________________________(1/1)
forall A : Prop, A -> A
```

- Enter command "Intros" (system uses the intro rule backwards, twice):

```
1 subgoal
A : Prop
H : A
____________________________________(1/1)
A
```

- Enter command "Assumption":

```
No more subgoals.
```

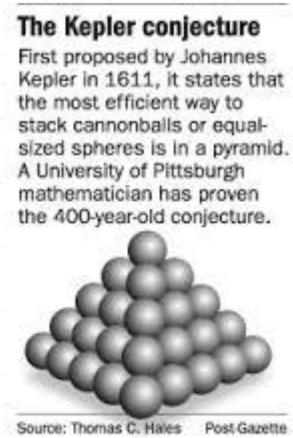
- Enter command "Qed":

```
tautology is defined
```

# MTT-based technology and applications (recap)

- Proof technology based on type theories
  - Proof assistants
    - MTT-based: ALF/Agda, Coq, Lean, Lego, NuPRL, Plastic, …
    - HOL-based: Isabelle, Isabelle-HOL, …
- Applications of proof assistants
  - Math: formalisation of mathematics – eg,
    - 4-colour theorem (on map colouring) in Coq
    - Kepler conjecture (on sphere packing) in Isabelle/HOL
  - Computer Science:
    - Program verification and advanced programming
  - Computational Linguistics
    - NL reasoning based on MTT-semantics

      (In Coq: Chatzikyriakidis & Luo 2014/2016/2020; Luo 2023)



# NL Reasoning in Coq

- Proof assistant Coq (INRIA, France (Coq 2004))
- Some basic data in MTT-semantics in Coq

```
(*  CNs as types *)
Definition CN := Set.
Parameters Animal Cat Elephant Human Obj: CN.
Parameters John Julie : Human.

(* coercive subtyping relations *)
Axiom ca : Cat -> Animal.           Coercion ca : Cat >-> Animal.
Axiom ea : Elephant -> Animal.  Coercion ea : Elephant >-> Animal.
Axiom ao : Animal -> Obj.           Coercion ao : Animal >-> Obj.
```

# Adjectival modification (intersective: black)

```
(* intersective adjective (black) *)
Parameter black : Obj -> Prop.

(* In Coq, "Record" types are Sigma-types *)
Record BCat := mkBC
        { cat :> Cat;
          pBlack : black(cat)
        }.

(* Any black cat is black. *)
Theorem bcat_is_black : forall bc : BCat, black(bc).
intros. apply bc.
Qed.    (* After Qed, bcat_is_black becomes the name of the proof. *)
```

- Further information, including other simple formalisations mentioned in the lectures
  - Adjective modifications (subsective, privative, …)
  - Donkey anaphora (and Most)
  - Dependant event types (e.g., EQP, selection restriction, ...)

can be found in (Luo 2023, Chap 5, esp. Sect 5.3)

# V.2. Dependent Categorial Grammar

- Categorial Grammars (or type-logical grammars)
  - An approach to syntactic analysis
  - CGs are based on substructural logics
    - Moortgat: 'Typelogical grammars are substructural logics, designed for reasoning about the composition of form and meaning in natural language.' (Stanford Encyclopedia of Philosophy, 2010)
- What is a substructural logic?
  - In a proof system, there are three kinds of "structural" rules:

    (1) Weakening: adding more assumptions

    (2) Contraction (strengthening): removing repeated/unused assumptions

    (3) Exchange: swapping the order of two assumptions

  In substructural (resource-sensitive) logics, the above may not be OK.

  In Lambek/CGs, none of them is OK.

# Lambek calculus and beyond

- Historical developments:
  - Ajdukiewicz, Bar-Hillel, …
- Lambek calculus (1958)
  - Ordered formulae B/A and A\B
    - John runs – "run applies to a np on the left".
      John : NP and run : NP\S
  - Resource sensitive
    - A context $\Gamma$, standing for a sequence of words, represents a sentence if $\Gamma$ |- S .
    - Words in a sentence cannot be arbitrarily added/removed/swapped
      ➔ context restrictions
      ➔ substructural logics

# An example

(*) John runs quickly.

|  | category (type) |
|---|---|
| John | NP |
| runs | NP \ S |
| quickly | (NP \ S) \ (NP \ S) |

We have, corresponding to (*):

NP, NP\S, (NP\S)\(NP\S) |- S

As the following derivation shows:

$$\dfrac{np \qquad \dfrac{(np \backslash S) \qquad (np \backslash S) \backslash (np \backslash S)}{(np \backslash S)} \backslash_e}{S} \backslash_e$$

- 1958 → … → 1980s … (CGs further developed)
  - Key: nice account of syntax/semantics interface – close correspondence between CGs and Montague semantics:
    [S] = **t** [NP] = **e** [CN] = **e**→**t** [A\B] = [B/A] = A→B
- Further (more recent) developments includes
  - Linear CGs (Girard's linear logic; 1987)
    - (Oehrle 1994) to initiate, among many others
  - Hybrid CGs (combining ordered/linear types)
    - For example: Kubota & Levine's HTLG (a recent book in 2020), among others

# Substructural type theory $\bar{\lambda}_\Pi$

- Linear types/terms:

| | $\Pi$-type | Non-dependent type | Abstraction | Application |
|---|---|---|---|---|
| Linear | $\overline{\Pi} x{:}A.B$ | $A \multimap B$ | $\bar{\lambda} x{:}A.b$ | $\overline{app}(f, a)$ |
| Ordered (right) | $\Pi^r x{:}A.B$ | $B / A$ | $\lambda^r x{:}A.b$ | $app^r(f, a)$ |
| Ordered (left) | $\Pi^l x{:}A.B$ | $A \setminus B$ | $\lambda^l x{:}A.b$ | $app^l(a, f)$ |

**Table 1** Three substructural function types in $\bar{\lambda}_\Pi$: summary of notations.

- Terms, rather than contexts, represent NL phrases.

Work based on (Luo 2015, Luo & Zhang 2016; see Luo 2023)

# Rules for the system without dependent types

*Variables*

$$\frac{}{x{:}A \vdash x : A}$$

*Ordered function types*

$$\frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda^r x{:}A.b : B/A} \qquad \frac{\Gamma \vdash f : B/A \quad \Delta \vdash a : A \quad FV(\Gamma) \cap FV(\Delta) = \emptyset}{\Gamma, \Delta \vdash app^r(f, a) : B}$$

$$\frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda^l x{:}A.b : A \setminus B} \qquad \frac{\Gamma \vdash f : A \setminus B \quad \Delta \vdash a : A \quad FV(\Gamma) \cap FV(\Delta) = \emptyset}{\Delta, \Gamma \vdash app^l(a, f) : B}$$

*Linear function types*

$$\frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \overline{\lambda} x{:}A.b : A \multimap B} \qquad \frac{\Gamma \vdash f : A \multimap B \quad \Delta \vdash a : A \quad FV(\Gamma) \cap FV(\Delta) = \emptyset}{\Gamma, \Delta \vdash \overline{app}(f, a) : B}$$

*The lexicon rule*

$$\frac{(c, A) \in \textsc{Lex}}{\langle\rangle \vdash c : A}$$

- Notes: if there is no dependent type, types can be defined first/independently:

**Definition 1 (types in $\bar{\lambda}_{\rightarrow}$)** *Types in $\bar{\lambda}_{\rightarrow}$ are inductively defined as follows:*

1. *The basic categories (such as* S, NP *and* CN*) are types.*
2. *If $A$ and $B$ are types, so are $A \multimap B$, $B/A$ and $A \backslash B$.* □

# An example without dep types (c.f., earlier example)

**Table 2** Lexicon for (8).

| | **category (type)** |
|---|---|
| John | NP |
| runs | NP \ S |
| quickly | (NP \ S) \ (NP \ S) |

(8) John runs quickly.

The lexicon for (8) is given in Table 2. It is straightforward to have:

$$app^l(John, app^l(runs, quickly)) : \mathrm{S}$$

When applying function $\phi$ to the above term, we have:

$$\phi(app^l(John, app^l(runs, quickly))) = John \circ runs \circ quickly$$

# A "counter-example"

(14) (#) a very book

Table 4 Lexicon for (14).

| | category (type) |
|---|---|
| a | NP/CN |
| very | (CN/CN)/(CN/CN) |
| book | CN |

- Example from (Moot & Retore 2012)
- In Lambek, we'd need a side condition for (/-intro) – context's non-emptiness.
- Otherwise, (14) would be a legitimate phrase:

$$a : \mathrm{NP/CN},\ \mathit{very} : (\mathrm{CN/CN})/(\mathrm{CN/CN}),\ \mathit{book} : \mathrm{CN}\ \vdash\ \mathrm{NP}$$

- In our setting, we have

$$\mathit{app}^r(a, \mathit{app}^r(\mathit{app}^r(\mathit{very}, \lambda^r x{:}\mathrm{CN}.x), \mathit{book}))\ :\ \mathrm{NP}$$

but this term does not represent a legitimate phrase (the λ-term blocks it!)

# Rules for substructural Π-types

*Formation rules for substructural Π-types*

$$\frac{\Gamma,\ x{:}A \vdash B\ type}{\Gamma \vdash \Pi^r x{:}A.B\ type} \qquad \frac{\Gamma,\ x{:}A \vdash B\ type}{\Gamma \vdash \Pi^l x{:}A.B\ type} \qquad \frac{\Gamma,\ x{:}A \vdash B\ type}{\Gamma \vdash \overline{\Pi} x{:}A.B\ type}$$

*Ordered Π-types* (($app^r$) and ($app^l$) have the side condition $FV(\Gamma) \cap FV(\Delta) = \emptyset$.)

$$(\lambda^r)\ \frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda^r x{:}A.b : \Pi^r x{:}A.B} \qquad (app^r)\ \frac{\Gamma \vdash f : \Pi^r x{:}A.B \quad \Delta \vdash a : A}{\Gamma, \Delta \vdash app^r(f, a) : [a/x]B}$$

$$(\lambda^l)\ \frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda^l x{:}A.b : \Pi^l x{:}A.B} \qquad (app^l)\ \frac{\Gamma \vdash f : \Pi^l x{:}A.B \quad \Delta \vdash a : A}{\Delta, \Gamma \vdash app^l(a, f) : [a/x]B}$$

*Linear Π-types* (($\overline{app}$) has the side condition $FV(\Gamma) \cap FV(\Delta) = \emptyset$.)

$$(\overline{\lambda})\ \frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \overline{\lambda} x{:}A.b : \overline{\Pi} x{:}A.B} \qquad (\overline{app})\ \frac{\Gamma \vdash f : \overline{\Pi} x{:}A.B \quad \Delta \vdash a : A}{\Gamma, \Delta \vdash \overline{app}(f, a) : [a/x]B}$$

# An example with dependent types

(23) Most students study hard.

Table 5 Lexicon for (23)

| | category (type) |
|---|---|
| most | $\Pi^r X{:}\mathrm{CN}.\ \mathrm{S}/(X \setminus \mathrm{S})$ |
| students | CN |
| study | NP \ S |
| hard | (NP \ S) \ (NP \ S) |

- In our system, we have

$$app^r(app^r(most, students), app^l(study, hard)) : \mathrm{S}$$

$$\phi(app^r(app^r(most, students), app^l(study, hard)))$$
$$= \mathrm{most} \circ students \circ study \circ hard$$

So, (23) is a legitimate sentence.

# Linearity

*Variables*

$$(Var) \qquad \frac{\Gamma, x{:}A \ \mathit{valid}}{\Gamma, x{:}A \vdash x{:}A} \quad (\forall y \in FV(\Gamma).\ x \sim_{\Gamma,x:A} y)$$

*where, in the side condition of* $(Var)$*, for any* $\Delta = x_1{:}A_1, ..., x_n{:}A_n$*, the dependency relation* $\sim_\Delta$ *is defined as: (1) if* $y \in FV(A_i)$*, then* $x_i \sim_\Delta y$*; (2) if* $x \sim_\Delta y$ *and* $y \sim_\Delta z$*, then* $x \sim_\Delta z$*.*

- Theorem (linearity)

*(Weak linearity in* $\bar{\lambda}_\Pi$*) In* $\bar{\lambda}_\Pi$*, every contextual variable occurs free essentially for exactly once in a well-typed term. In symbols, if* $\Gamma \vdash a : A$ *with* $\Gamma = x_1{:}A_1, ..., x_n{:}A_n$*, then each* $x_i$ *occurs free essentially in* $a$ *for exactly once (i.e.,* $x_i \in E_\Gamma(a)$ *for exactly once* $(i = 1, ..., n)$*).*