International Conference on Mathematical and Computational Linguistics for Proofs
17th September 2025, Institut Pascal, Orsay, France

Natural language understanding in natural proof checking

Adrian De Lon [1, 2]

[1] Mathematical Logic Group, University of Bonn
[2] CIIRC CTU, Prague
Supported by GAČR Grant 25-17929X *NextReason*

CZECH INSTITUTE
OF INFORMATICS
ROBOTICS AND
CYBERNETICS
CTU IN PRAGUE

Formalization examples

Example: Cantor's theorem formalized in Naproche-ZF

Theorem (Cantor). There exists no surjection from A to 2^A.
Proof. Suppose not. Consider a surjection f from A to 2^A. Let $B = \{a \in A \mid a \notin f(a)\}$. Then $B \in 2^A$.
There exists $a' \in A$ such that $f(a') = B$. Now $a' \in B$ iff $a' \notin f(a') = B$. Contradiction.

Example: proof tasks from proof gaps

Theorem (Cantor). There exists no surjection from A to 2^A.
Proof. Suppose not. Consider a surjection f from A to 2^A. Let $B = \{a \in A \mid a \notin f(a)\}$. Then $B \in 2^A$. There exists $a' \in A$ such that $f(a') = B$. Now $a' \in B$ iff $a' \notin f(a') = B$. Contradiction.

Negated conjecture from "Then $B \in 2^A$"	$B \notin 2^A$
Global premise from a lemma	$\forall X.\, \forall Y.\, X \subseteq Y \implies X \in 2^Y$
Global premise from a definition	$\forall X. \forall X.\, X \subseteq Y \iff (\forall x.\, x \in X \implies x \in Y)$
$\vdots$	$\vdots$
Local premise from "Let $B = \cdots$"	$\forall a.\, a \in B \iff (a \in A \land a \notin f(a))$
Local premise from "Consider ..."	$f \in \mathsf{Surj}(A, 2^A)$
Local premise from "Suppose not"	$\exists g.\, g \in \mathsf{Surj}(A, 2^A)$

Example: proof tasks from proof gaps

Theorem (Cantor). There exists no surjection from A to 2^A.
Proof. Suppose not. Consider a surjection f from A to 2^A. Let $B = \{a \in A \mid a \notin f(a)\}$. Then $B \in 2^A$.
There exists $a' \in A$ such that $f(a') = B$. Now $a' \in B$ iff $a' \notin f(a') = B$. Contradiction.

Negated conjecture from "Then $B \in 2^A$"	$B \notin 2^A$
Global premise from a lemma	$\forall X.\, \forall Y.\, X \subseteq Y \implies X \in 2^Y$
Global premise from a definition	$\forall X. \forall X.\, X \subseteq Y \iff (\forall x.\, x \in X \implies x \in Y)$
$\vdots$	$\vdots$
Local premise from "Let $B = \cdots$"	$\forall a.\, a \in B \iff (a \in A \land a \notin f(a))$
Local premise from "Consider ..."	$f \in \mathrm{Surj}(A, 2^A)$
Local premise from "Suppose not"	$\exists g.\, g \in \mathrm{Surj}(A, 2^A)$

Example: proof tasks from proof gaps

Theorem (Cantor). There exists no surjection from A to 2^A.
Proof. Suppose not. Consider a surjection f from A to 2^A. Let $B = \{a \in A \mid a \notin f(a)\}$. Then $B \in 2^A$. There exists $a' \in A$ such that $f(a') = B$. Now $a' \in B$ iff $a' \notin f(a') = B$. Contradiction.

Negated conjecture from "Then $B \in 2^A$"	$B \notin 2^A$
Global premise from a lemma	$\forall X. \forall Y. X \subseteq Y \implies X \in 2^Y$
Global premise from a definition	$\forall X. \forall X. X \subseteq Y \iff (\forall x. x \in X \implies x \in Y)$
$\vdots$	$\vdots$
Local premise from "Let $B = \cdots$"	$\forall a. a \in B \iff (a \in A \wedge a \notin f(a))$
Local premise from "Consider ..."	$f \in \mathrm{Surj}(A, 2^A)$
Local premise from "Suppose not"	$\exists g. g \in \mathrm{Surj}(A, 2^A)$

Example: formalizing in natural language with markup

Theorem (Burali-Forti antimony) There exists no set Ω such that for all α we have $\alpha \in \Omega$ iff α is an ordinal.

Proof. Suppose not. Consider Ω such that for all α we have $\alpha \in \Omega$ iff α is an ordinal. For all x, y such that $x \in y \in \Omega$ we have $x \in \Omega$. So Ω is $\in$-transitive. Thus Ω is an ordinal. Hence $\Omega \in \Omega$. Contradiction. $\square$

```
\begin{theorem}[Burali-Forti antimony]\label{burali_forti}
    There exists no set $\Omega$ such that
    for all $\alpha$ we have $\alpha\in \Omega$ iff $\alpha$ is an ordinal.
\end{theorem}
\begin{proof}
    Suppose not.
    Consider $\Omega$ such that for all $\alpha$ we have
        $\alpha\in \Omega$ iff $\alpha$ is an ordinal.
    For all $x, y$ such that $x\in y\in\Omega$ we have $x\in\Omega$.
    So $\Omega$ is \in-transitive. Thus $\Omega$ is an ordinal.
    Hence $\Omega\in\Omega$. Contradiction.
\end{proof}
```

Example: phase transition to controlled natural language

Informal statement from T. Jech, *Set Theory*, Ex. 24.3:
If $2^{\aleph_\alpha} \leq \aleph_{\alpha+2}$ holds for all cardinals of cofinality ω,
then the same holds for all singular cardinals.

Formalized statement in controlled language:
If $2^{\aleph_\alpha} \leq \aleph_{\alpha+2}$ for all cardinals α of cofinality ω,
then $2^{\aleph_\beta} \leq \aleph_{\beta+2}$ for all singular cardinals β.

Formal translation to first-order form:

$$(\forall \alpha.\, \mathrm{Card}(\alpha) \land \mathrm{cf}(\alpha) = \omega \rightarrow 2^{\aleph_\alpha} \leq \aleph_{\alpha+2})$$
$$\rightarrow (\forall \beta.\, \mathrm{Sing}(\beta) \rightarrow 2^{\aleph_\beta} \leq \aleph_{\beta+2})$$

How did we get here?
Where are we going?

This dissertation describes some investigations into the possible use of a digital computer to ***check mathematical proofs of the type that normally appear in textbooks****. A computer program called the Proofchecker was written that verifies proofs written in a specified input format. A two-step process is involved in checking a proof within this framework: the proof is first translated from the language of the textbook proof into the input language of the Proofchecker, and the Proofchecker then attempts to translate the input proof into a rigorous proof, i.e., into a sequence of steps in a formal logical system.*

P W Abrahams, *Machine Verification of Mathematical Proof* (1963)

The original aim of the writer was to ***take mathematical textbooks*** *such as Landau on the number system, Hardy–Wright on number theory, Hardy on the calculus, Veblen–Young on projective geometry, the volumes by Bourbaki,* ***as outlines and to make the machine formalize all the proofs*** *(fill in the gaps),*

Hao Wang, *Toward Mechanical Mathematics* (1960)

A biased and incomplete history

1960	resolution, advent of automated theorem proving (Davis, Putnam, et al.)
1961–1963	Proofchecker, attempt to check textbooks proofs in a Lisp representation (Abrahams)
1967	Automath, influential for later proof assistants (de Bruijn)
1973–now	Mizar, quasinatural language, various iterations (Trybulec et al.)
1970	Evidence Algorithm ambitions, first attempts (Glushkov et al.)
1998–now	Grammatical Framework (Ranta et al.)
2002–2008	SAD as realization of PC/EA dreams, but still at a small scale (Paskevich et al.)
2003–2014	Naproche and earlier related projects (Koepke, Schröder, Cramer et al.)
2018–now	Naproche-SAD, scaling SAD to chapter size (Koepke et al.)
2023–now	Naproche-ZF, scaling a bit further?
2025–now	MALINCA, incl. Godement challenge

Example formalization in SAD

```
Theorem Tarski.
    Let U be a complete lattice and f be an monotone function on U.
    Let S be the set of fixed points of f. Then S is a complete lattice.
Proof.
    Let T be a subset of S.
    Let us show that T has a supremum in S.
        Take P = { x << U | f(x) <= x and x is an upper bound of T in U }.
        Take an infimum p of P in U.
        f(p) is a lower bound of P in U and an upper bound of T in U.
        Hence p is a fixed point of f and a supremum of T in S.
    End.
    Let us show that T has an infimum in S.
        Take Q = { x << U | x <= f(x) and x is a lower bound of T in U }.
        Take a supremum q of Q in U.
        f(q) is an upper bound of Q in U and a lower bound of T in U.
        Hence q is a fixed point of f and an infimum of T in S.
    End.
QED.
```

A Proofchecker example revisited

Original	Therefore, there exists an element a' such that $aa' = e$.
Proofchecker	`(IMPLIES (IDENT E G) (EXISTS A1 (EQUAL (GMULT A1 A) E)))`
Mizar	`thus ex a' being Element of G st a * a' = 1.G;`
SAD	`Thus there exists an element a' of G such that a * a' = 1.`
Naproche-ZF	`Therefore, there exists an element $a'$ such that $a\mul a' = \one$.`

A Proofchecker example revisited

Original	Therefore, there exists an element a' such that $aa' = e$.
Proofchecker	`(IMPLIES (IDENT E G) (EXISTS A1 (EQUAL (GMULT A1 A) E)))`
Mizar	`thus ex a' being Element of G st a * a' = 1.G;`
SAD	`Thus there exists an element a' of G such that a * a' = 1.`
Naproche-ZF	`Therefore, there exists an element $a'$ such that $a\mul a' = \one$.`
	Therefore, there exists an element a' such that $aa' = e$.

A Proofchecker example revisited

Original	Therefore, there exists an element a' such that $aa' = e$.
Proofchecker	`(IMPLIES (IDENT E G) (EXISTS A1 (EQUAL (GMULT A1 A) E)))`
Mizar	`thus ex a' being Element of G st a * a' = 1.G;`
SAD	`Thus there exists an element a' of G such that a * a' = 1.`
Naproche-ZF	`Therefore, there exists an element $a'$ such that $a\mul a' = \one$.`
	Therefore, there exists an element a' such that $aa' = e$.

(“`$a$ is an element`” can be defined as an abbreviation of `$a\in\carrier$`, where `\carrier[G]` is the carrier of the group, with the bracketed argument optionally inferred)

Natural language understanding and proof checking

Syntactic checking	Is the text grammatical? Is the vocabulary defined?
Tools	tokenizer, scanner, parser
Result	token stream, lexicon, syntax tree
Logical checking	Does each step follow from the steps before?
Tools	automated theorem prover, maybe also equational provers and specialized solvers
Result	truth value/proof trace/proof object
Ontological checking	Does the text make sense? Are symbols welldefined at use?
Tools	automated theorem prover, type checker, model finder
Result	*you're not even wrong*: inconsistency of axioms, type derivations, countermodels

Tokenization

Using stateful combinators, to correctly handle differences between math and text mode.

$\sin$ (`$\sin$` or `$\operatorname{sin}$`) vs sin (`$sin$`)

Nesting of math and text with `\text{...}`, &c.

What about symbols as part of text-level lexical items? E.g., $\in$-induction, C*-algebra?

Naproche-ZF extracts token patterns of lexical items from definitions. We can use the context of the definition to distinguish between nouns, verbs, adjectives, &c.

Smart paradigms à la GF (Grammatical Framework) are used to guess plural forms of nouns and verbs.

Use of semantic macros is encouraged.

Examples:

"f is a function from X to Y iff …"
$\rightsquigarrow$ noun: function[/s] $(-_0)$ from $(-_1)$ to $(-_2)$

"A is a matrix over k iff …"
$\rightsquigarrow$ noun: matri[x/ces] $(-_0)$ over $(-_1)$

"x is δ-close iff …"
$\rightsquigarrow$ adjective: $(-_1)$-close

"m divide[s/] n iff …"
$\rightsquigarrow$ verb: divides $(-_1)$

"$m < n$ iff …"
$\rightsquigarrow$ relation symbol: $<$

"$x \cup y = \cdots$"
$\rightsquigarrow$ infix function symbol: $\cup$

Grammar-oriented approach to NLU: grammar fragment parametrized by lexical items

Dynamic: patterns for lexical items

noun $\longrightarrow$ set | group | function from *term* to *term* | *term*-ary relation | $\cdots$

mixfix-operator $\longrightarrow$ $\mathit{expr} + \mathit{expr} \mid \bigcup \mathit{expr} \mid \mathit{expr}\,! \mid \langle \mathit{expr}, \mathit{expr} \rangle \mid \cdots$

adjective $\longrightarrow$ even | continuous | *term*-close | (*expr*, *expr*)-provable | $\cdots$

relator $\longrightarrow$ $= \mid \in \mid < \mid \cong_{\mathit{expr}} \mid \cdots$

Static: phrases, sentences, blocks

noun-phrase $\longrightarrow$ *adjective-list noun attribute such-that-statement*

such-that-statement $\longrightarrow$ such that *statement* | ε

statement $\longrightarrow$ *not* | *if* | *iff* | *xor* | *nor* | *exists* | *quantified-phrase* | $\cdots$

atomic-statement $\longrightarrow$ *formula* | *noun-statement* | *verb-statement* | *adjective-statement* | $\cdots$

noun-statement $\longrightarrow$ *term* is a *noun-phrase* | *term* is not a *noun-phrase*

let $\longrightarrow$ let *var* be a *noun phrase*. | let *var-list* $\in$ *expression*.

assumption-list $\longrightarrow$ suppose *statement*. *assumption-list* | *let assumption-list* | ε

theorem $\longrightarrow$ *assumption-list statement*.

Examples: basic syntactic transformation

$a, b < c < d$
$\rightsquigarrow a < c \wedge b < c \wedge c < d$

$x \; R \; y$
$\rightsquigarrow (x, y) \in R$

For all $a < b$ such that $P(a)$ we have $Q(a)$.
$\rightsquigarrow$ For all a we have if $a < b$ and $P(a)$, then $Q(a)$.

There exists x such that ...
$\rightsquigarrow$ Consider x such that ....

Every set is an element of some Grothendieck universe.
$\rightsquigarrow$ For all sets x there exists a Grothendieck universe U such that x is an element of U.

Remove ambiguities from the language by distinguishing math and text mode (e.g. variable “a” vs article “a”) and by enforcing number agreement.

Earley’s algorithm guarantees better asymptotic behaviour than backtracking monadic parser combinators (cubic vs. exponential).

Declarative grammar specification is easier to extend.

Only content of *formal environments* such as `definition`, `theorem`, and `proof` is checked by the system. Everything else is treated as informal commentary.

One can freely mix informal and formal material in the same document.

The formal material is already readable and does not need to be restated in informal language (less clutter and no issues with syncing).

Natural proof vernacular

What counts as a proof? Some (in)famous one-liners

"I have discovered a truly marvellous proof of this, which this margin is too narrow to contain."

"Left as an exercise to the reader."

"Follows by induction."

"Follows from an easy diagram chase."

"Easy (using 1.5 of course)."

"Check (part 3 is like 3.6)."

"Think."

Proof steps for humans vs. computers

Things that are perhaps not obvious to computers:
proofs by analogy, proofs using geometric intuition, proofs using *inventive devices*, &c.

Obvious to computers:
decidable theories, checking 10000 easy cases, TPTP problems with a low difficulty rating, &c.

Automated theorem provers don't need to think like humans, they just need to be good enough.

Still hard to get 100% coverage for typical human-sized proof steps (long tail problem).

We can use a portfolio approach for better coverage.

Proof rules in Naproche-ZF: some terminals

Γ is the set of global premises, containing all previous theorems and definitions.

Λ is the set of local premises, containing local assumptions and claims from previous proof steps.

Γ^{sel} is a subset of Γ after optional premise selection.

$$\frac{\Gamma^{\text{sel}}; \Lambda \vdash^{\text{ATP}} \varphi}{\Gamma; \Lambda \vdash \varphi} \ \square$$

$$\frac{\Lambda, \gamma_1, \ldots, \gamma_k \vdash^{\text{ATP}} \varphi}{\Gamma; \Lambda \vdash \varphi} \ \text{by } \gamma_1, \ldots, \gamma_k \in \Gamma$$

$$\frac{\Lambda \vdash^{\text{ATP}} \varphi}{\Gamma; \Lambda \vdash \varphi} \ \text{by assumption}$$

$$\frac{\Gamma^{\text{sel}}; \Lambda \vdash^{\text{ATP}} \forall x.\, x \in X \longleftrightarrow x \in Y}{\Gamma; \Lambda \vdash X = Y} \ \text{setext}$$

Proof rules in Naproche-ZF: some intermediate steps

$$\frac{\Gamma; \Lambda, \varphi \vdash \psi \qquad \Gamma^{\mathsf{sel}}; \Lambda \vdash^{\mathsf{ATP}} \varphi}{\Gamma; \Lambda \vdash \psi}\ \text{have } \varphi \qquad \frac{\Gamma; \Lambda \vdash \varphi \qquad \Gamma^{\mathsf{sel}}; \Lambda, \varphi \vdash^{\mathsf{ATP}} \psi}{\Gamma; \Lambda \vdash \psi}\ \text{suffices } \varphi$$

$$\frac{\Gamma; \Lambda, \varphi \vdash \psi}{\Gamma; \Lambda \vdash \varphi \to \psi}\ \text{assume } \varphi \qquad \frac{\Gamma; \Lambda, \varphi \vdash \bot}{\Gamma; \Lambda \vdash \neg\varphi}\ \text{assume } \varphi \qquad \frac{\Gamma; \Lambda, \neg\varphi \vdash \bot}{\Gamma; \Lambda \vdash \varphi}\ \text{suppose not}$$

$$\frac{\Gamma; \Lambda, \varphi_1 \vdash \psi \quad \cdots \quad \Gamma; \Lambda, \varphi_k \vdash \psi \qquad \Gamma^{\mathsf{sel}}; \Lambda \vdash^{\mathsf{ATP}} \bigvee_i \varphi_i}{\Gamma; \Lambda \vdash \psi}\ \text{cases } \varphi_1, \ldots, \varphi_k$$

$$\frac{\Gamma; \Lambda, a \vdash \varphi(a)}{\Gamma; \Lambda \vdash \forall a. \varphi(a)}\ \text{let } a \qquad \frac{\Gamma \vdash \forall a. (\forall b. b \in a \to \varphi(b)) \to \varphi(a)}{\Gamma; \Lambda \vdash \forall c. \varphi(c)}\ \in\text{-induction}$$

$$\frac{\Gamma; \Lambda, a, \psi(a) \vdash \varphi \qquad \Gamma^{\mathsf{sel}}; \Lambda \vdash^{\mathsf{ATP}} \exists a. \psi(a)}{\Gamma; \Lambda \vdash \varphi}\ \text{consider } a \text{ such that } \psi(a)$$

Transparency is important when things don't go smoothly.

The translation to formal logic should be straightforward/unsurprising (e.g. it's better to report an ambiguity error than to arbitrarily disambiguate).

The resulting formulas should be easy to inspect (no more names like "`zpzlzuzs`").

Hard problem: how to present ATP proofs to the user?

Autoformalization and informalization with Mizar and Naproche

Joint work with Mario Carneiro, Atle Hahn, Peter Koepke, and Josef Urban

Informalization and proof compression

```
definition let X,Y;
    pred X c= Y means for x being object holds x in X implies x in Y;
```

Definition. Let X, Y be sets. $X \subseteq Y$ iff for all objects x such that $x \in X$ we have $x \in Y$.

```
theorem for C being countable Language, phi wff string of C, X being set st
    X c= AllFormulasOf C & phi is X-implied holds phi is X-provable
```

Theorem (Completeness Theorem). Let L be a countable language, φ a wellformed L-formula, and Γ a set of L-formulas such that $\Gamma \vDash \varphi$. Then $\Gamma \vdash \varphi$.

```
theorem Th19:
  for T being non empty normal TopSpace, A,B being closed Subset of T st
  A <> {} & A misses B holds ex F being Function of T,R^1 st
  F is continuous & for x being Point of T holds 0 <= F.x & F.x <= 1 &
  (x in A implies F.x = 0) & (x in B implies F.x = 1)
```

Theorem (Urysohn). Let T be a non-empty normal space. Let A, B be closed subsets of T such that $A \neq \emptyset$ and $A \cap B = \emptyset$. Then there exists a continuous function f from T to $\mathbb{R}$ such that for all points x of T we have $0 \leq f(x) \leq 1$ and $x \in A \implies f(x) = 0$ and $x \in B \implies f(x) = 1$.

Why Mizar?

Large: MML is the largest quasinatural formal library with 1473 articles by over 260 authors, containing 3.6 M lines, 74 k theorems, and 15 k definitions.

Automatable: over 80% of problems from the MML can be solved by ATPs.

Set-theoretic: Mizar's Tarski–Grothendieck foundations are a stronger version of Zermelo–Fraenkel set theory which is the standard foundation of mathematics.

Pre-mapped: Journal of Formalized Mathematics publishing pipeline already includes a vocabulary mapping from Mizar identifiers to LaTeX-markup.

mizar-rs: a modern and performant reimplementation of the Mizar system by Mario

Our evil scheme (Part I): auto-informalization combining mizar-rs and Naproche-ZF

Obtain a vocabulary mapping from Mizar identifiers to patterns of LaTeX markup.

Implement a bidirectional verifiability-preserving syntactic translation from the Mizar language to controlled natural language, adapted from the front-end of Naproche-ZF.

Make the translated result more readable (with simple heuristics, LLMs, manual editing, &c.).

Give the theorems useful names and labels (instead of `Th01`,`Th02`,...).

The result should be understandable to mathematicians unfamiliar with Mizar or other proof assistants.

Vocabulary mapping (11k+) derived from the publishing process for Formalized Mathematics

(functor)	`[: A,B :]`	$A \times B$	(mixfix)
(functor)	`X "/\" Y`	$X \sqcap Y$	(mixfix)
(relation)	`A c= B`	$A \subseteq B$	(relation)
(relation)	`a,b equiv c,d`	$\overline{ab} \cong \overline{cd}$	(predicate)
(relation)	`f unifies t1,t2`	f unifies t_1 with t_2	(verb)
(relation)	`x is_/\-reducible_in X`	x is $\cap$-reducible in X	(adjective)
(mode)	`language of Y,S`	language over Y and S	(noun)
(mode)	`Homomorphism of G,H`	homomorphism from G to H	(noun)
(attribute)	`subst-forex`	$\forall$-$\exists$-substituting	(adjective)
(attribute)	`k-halting`	k-halting	(adjective)

Our evil scheme (Part II): proof automation and compression

Integrate ATPs into the Mizar checker.

Semi-automatically compress existing Mizar proofs to make their level of detail more natural.

Can we develop systematic criteria for *naturalness* of proofs?

Mario's mizar-rs can check the MML in under 3 minutes on a 128-thread CPU (most of the library finishes under a minute, but there are a couple of articles that are stragglers).

Grammar-based parser for the controlled language adds a bit of overhead compared to an optimized parser for the simpler quasinatural Mizar language.

We have to be smart about using ATPs and cache their results.

Autoformalization

Our evil scheme (Part III): LLM-based autoformalization in controlled natural language

Controlled natural language is a promising target for autoformalization, since LLMs have seen much more natural language mathematics in their training data than formal mathematics.

Restricting to controlled language via prompting works quite well.

Non-controlled sentences generated by the LLM can indicate where to extend the grammar.

One could use grammar augmentation (syntactically constrained sampling) to force controlled natural language output.

Other difficulties of autoformalization remain: global coherence, implicitness, high-level informal reasoning, &c.

Extract definitions from source and rewrite them into separate blocks, splitting definitions as needed.

Generate internal labels for each definition.

Each block introduces ones definition, which gets highlighted with `\emph{...}` or similar markup for easier processing.

Extract a separate lists of *well-known concepts* that are not explicitly defined in the source text and generate placeholder definitions for them.

Two-step approach to autoformalization: actual formalization

Formalize the preprocessed definitions with a cheap reasoning model, such as DeepSeek R1.

A large prompt constrains the language model to the controlled grammar.

Iterate on formalization attempts with a simple feedback loop with a syntax checker.

Initial autoformalization results: overview

We chose *inverse semigroups* as first informalization topic.

82% of proposed formalizations syntactically valid after 10 runs.

Initial autoformalization results: examples

```
\begin{definition}\label{def_idempotent_pure}
  Assume that $\theta$ is an homomorphism of inverse semigroups from $S$ to $T$.
  $\theta$ is idempotent-pure iff for all $s \in S_0$ if $\theta(s)$ is an
  idempotent in $T$ then $s$ is an idempotent in $S$.
\end{definition}
```

```
% Original version: A subset $I$ of a poset $(X, \le)$ is an \emph{order ideal}
%if whenever $ x \le y \in I $, then $ x \in I $.
\begin{definition}\label{def_order_ideal}
  $I$ is an order ideal of $P$ iff $P$ is a poset and there exist $X, R$ such that
  $P = (X, R)$ and for all $x, y \in X$ if $(x, y) \in R$ and $y \in I$ then $x \in I$.
\end{definition}
```

Thank you!