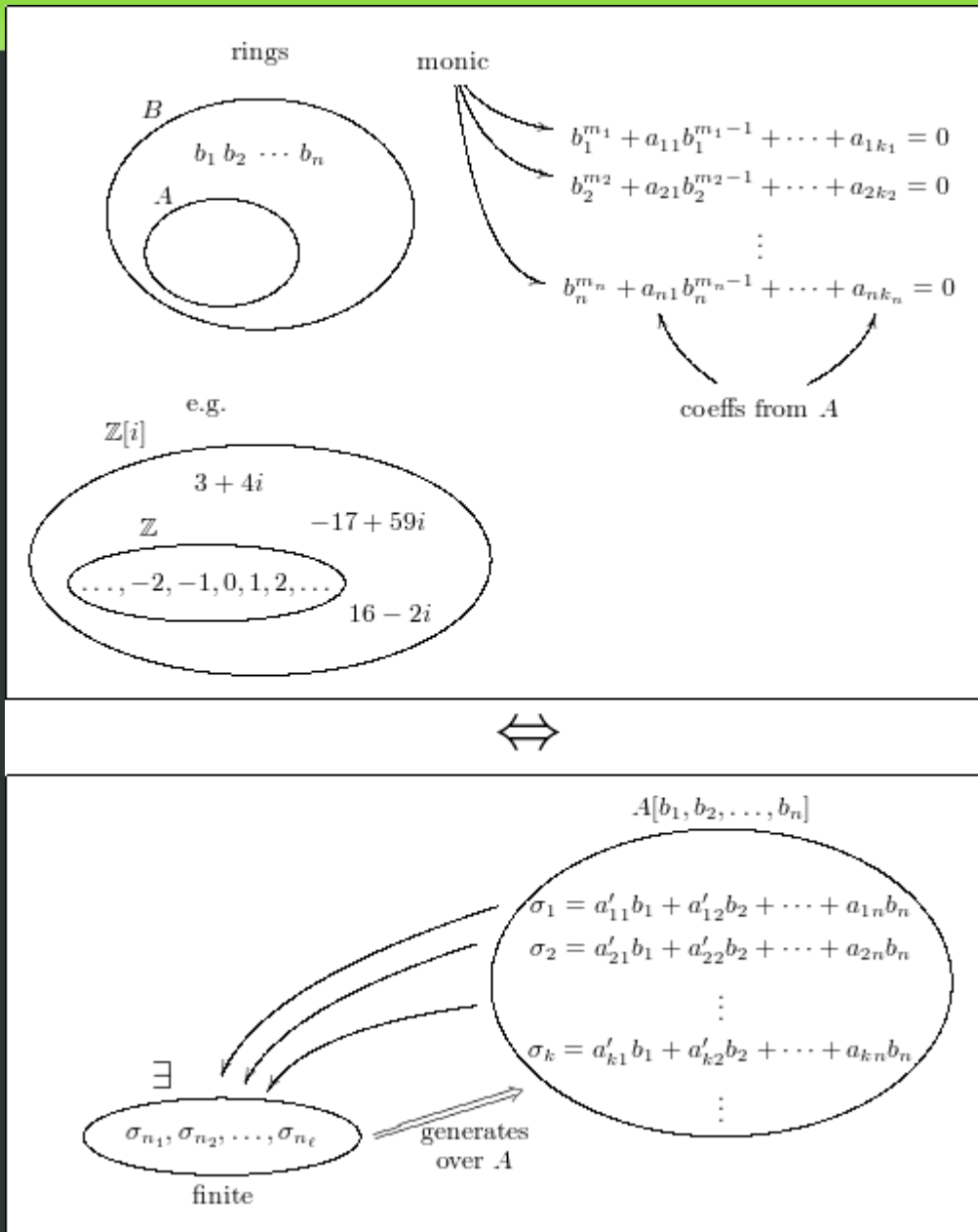


# Illustrating Mathematics



- Steve Kieffer
- Simon Fraser University

# The problem:

Transform this

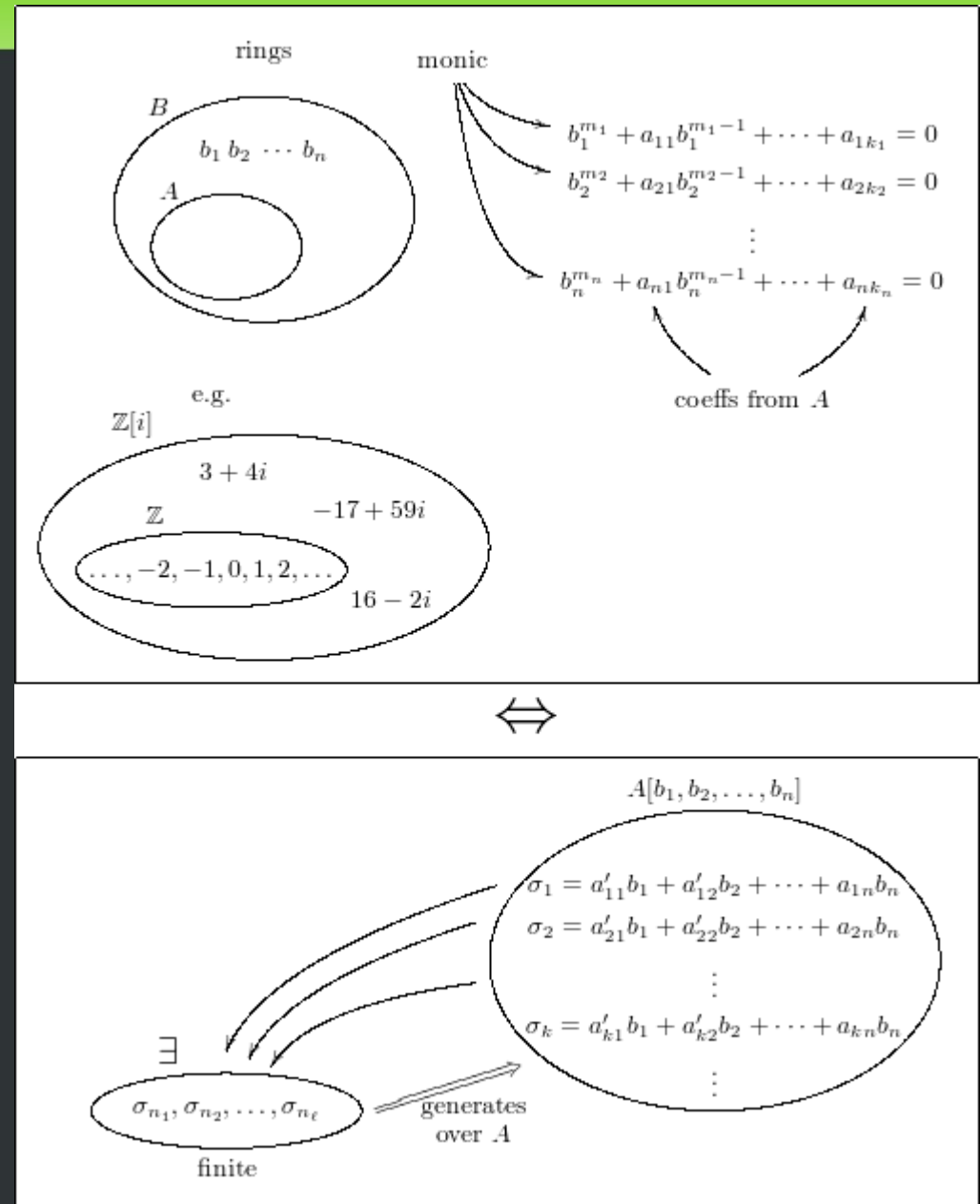


into this



Let  $A$  and  $B$  be rings, with  $A \subseteq B$ . Let finitely many elements  $b_1, \dots, b_n \in B$  be given. Then the  $b_i$  are all integral over  $A$  if and only if  $A[b_1, \dots, b_n]$  is a finitely generated  $A$ -module.

automatically.



# Deployment: GUI

The screenshot shows the Math Illustrator application window. The title bar reads "Math Illustrator". The menu bar includes "File", "Edit", and "Settings".

The main text area contains the following theorem:

Let  $A$  and  $B$  be rings, with  $A \subseteq B$ . Let finitely many elements  $b_1, \dots, b_n \in B$  be given. Then the  $b_i$  are all integral over  $A$  if and only if  $A[b_1, \dots, b_n]$  is a finitely generated  $A$ -module.

The diagram below illustrates the proof. On the left, under the heading "rings", two nested ovals represent rings  $A$  and  $B$ , with  $A \subseteq B$ . The elements  $b_1, b_2, \dots, b_n$  are shown inside  $B$ . Below this, under "e.g.", a diagram shows the ring  $\mathbb{Z}$  (represented by an oval containing  $\dots, -2, -1, 0, 1, 2, \dots$ ) as a subset of  $\mathbb{Z}[i]$  (represented by a larger oval containing  $3 + 4i$ ,  $-17 + 59i$ , and  $16 - 2i$ ).

On the right, under the heading "monic", three monic polynomials are shown:

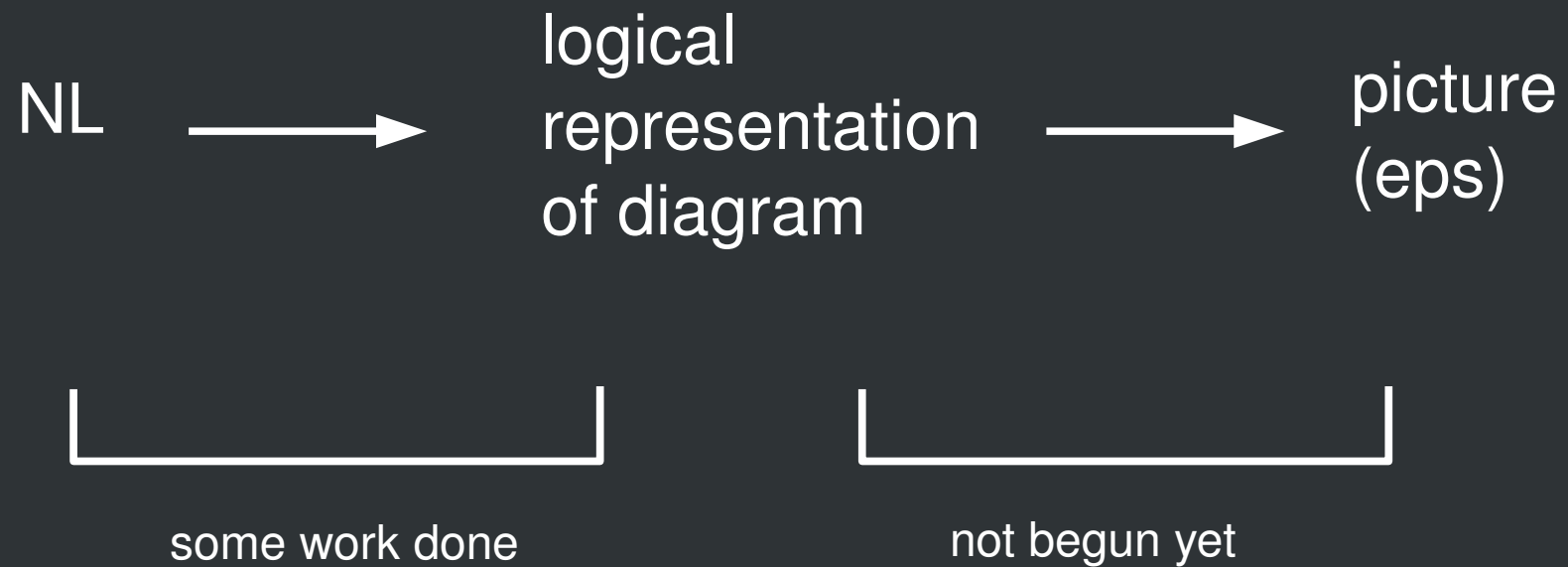
$$b_1 \mapsto x^{m_1} + a_{11}x^{m_1-1} + \dots + a_{1k_1}$$
$$b_2 \mapsto x^{m_2} + a_{21}x^{m_2-1} + \dots + a_{2k_2}$$
$$\vdots$$
$$b_n \mapsto x^{m_n} + a_{n1}x^{m_n-1} + \dots + a_{nk_n}$$

Arrows point from each  $b_i$  to its corresponding polynomial. A curved arrow labeled "coeffs from  $A$ " points from the coefficients  $a_{ij}$  back to the ring  $A$ .

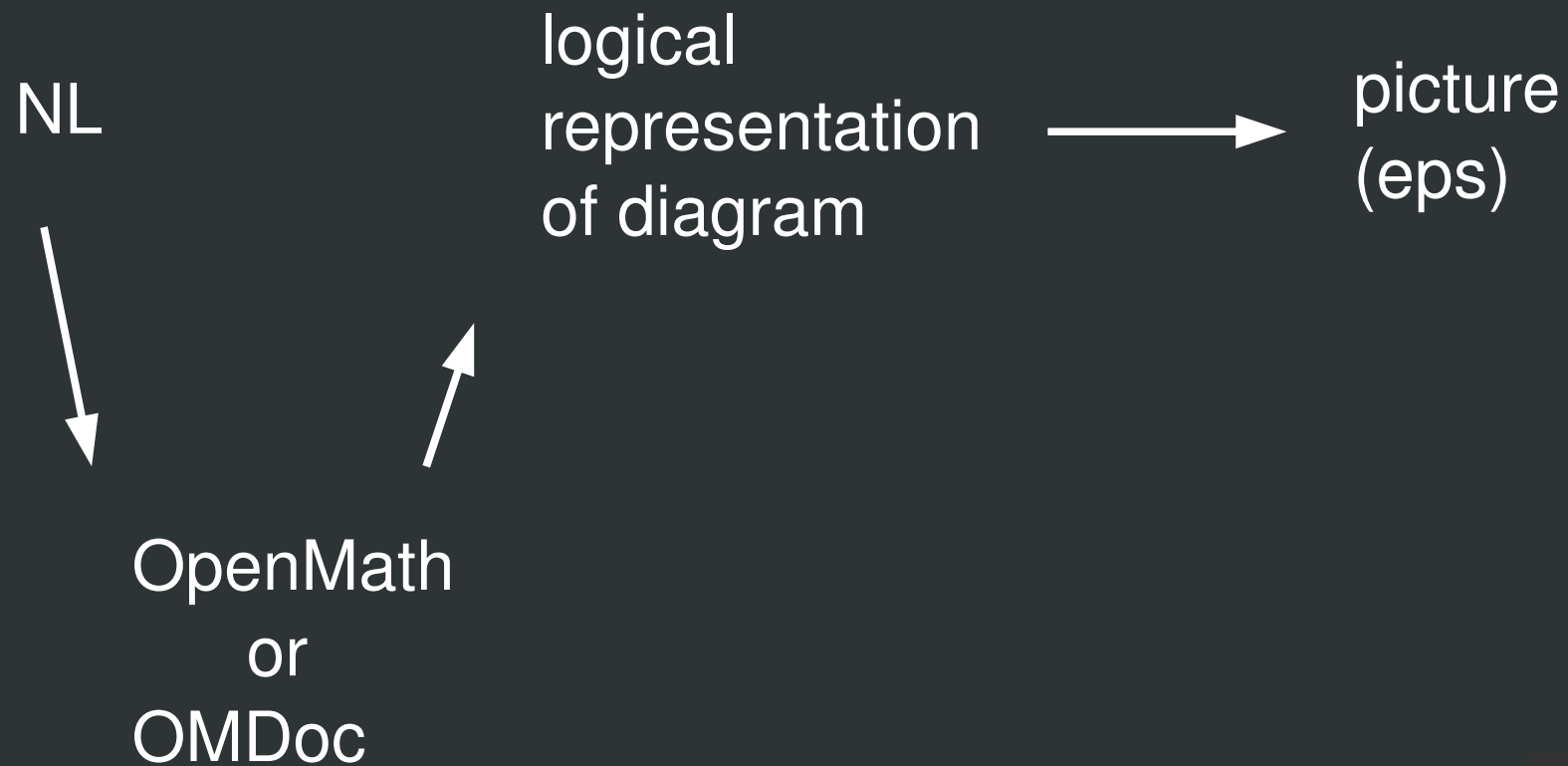
On the left side of the GUI, there are several dropdown menus for configuration:

- Group: kieffer1
- Ring: bruin1
- Field: bruin2
- Galois group: kieffer4
- Elliptic curve: bruin5

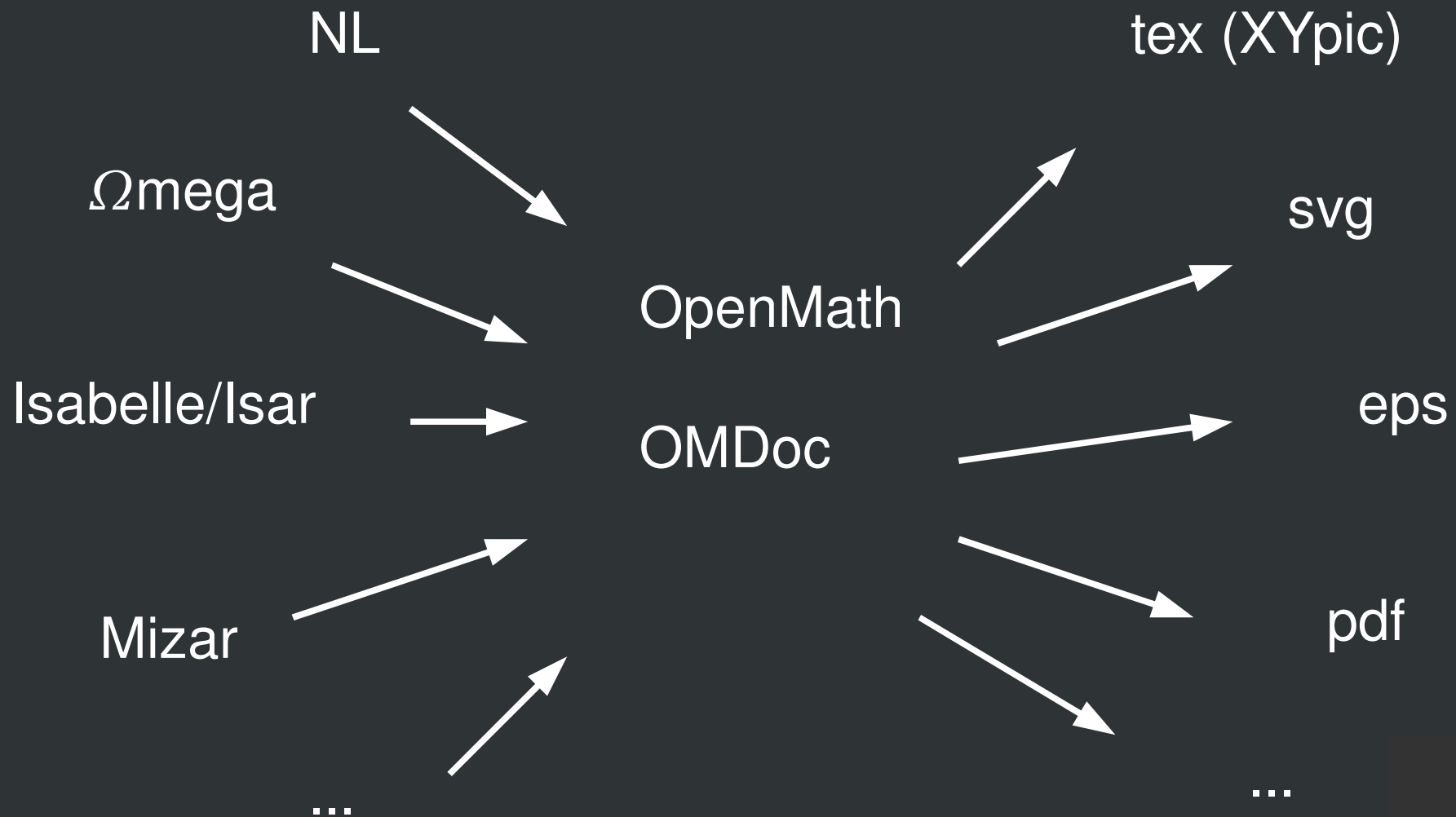
# Modularity



# Modularity, future



# Future generalization



# NL $\rightarrow$ LDR (logical diagram representation)

- Unit of input: a “paragraph”

Theorem statement in full:

Let  $A$  and  $B$  be rings, with  $A \subseteq B$ . Let finitely many elements  $b_1, \dots, b_n \in B$  be given. Then the  $b_i$  are all integral over  $A$  if and only if  $A[b_1, \dots, b_n]$  is a finitely generated  $A$ -module.

One paragraph in a proof?

Since  $\Omega = \bigcup_{n \geq 1} \mathbb{Q}(\mu_n)$ , we find  $G(\Omega|\mathbb{Q}) = \lim_{\leftarrow n} G(\mathbb{Q}(\mu_n)|\mathbb{Q}) = \lim_{\leftarrow n} (\mathbb{Z}/n\mathbb{Z})^* = \hat{\mathbb{Z}}^*$ . But  $\hat{\mathbb{Z}} = \prod_p \mathbb{Z}_p$ , and  $\mathbb{Z}_p^* \cong \mathbb{Z}_p \times \mathbb{Z}/(p-1)\mathbb{Z}$  for  $p \neq 2$  and  $\mathbb{Z}_2^* \cong \mathbb{Z}_2 \times \mathbb{Z}/2\mathbb{Z}$ . Consequently,  $G(\Omega|\mathbb{Q}) \cong \hat{\mathbb{Z}}^* \cong \hat{\mathbb{Z}} \times \hat{T}$ , where  $\hat{T} = \prod_{p \neq 2} \mathbb{Z}/(p-1)\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ . This shows that the torsion subgroup  $T$  of  $G(\Omega|\mathbb{Q})$  is isomorphic to the torsion sub-

# Thm lang. vs. Pf lang.

- Theorem language:
  - Let / Assume / Suppose ...
  - ... given
  - Then ...

Let  $A$  and  $B$  be rings, with  $A \subseteq B$ . Let finitely many elements  $b_1, \dots, b_n \in B$  be given. Then the  $b_i$  are all integral over  $A$  if and only if  $A[b_1, \dots, b_n]$  is a finitely generated  $A$ -module.



# Thm lang. vs. Pf lang.

Since  $\Omega = \cup_{n \geq 1} \mathbb{Q}(\mu_n)$ , we find

$$G(\Omega|\mathbb{Q}) = \lim_{\leftarrow n} G(\mathbb{Q}(\mu_n)|\mathbb{Q}) = \lim_{\leftarrow n} (\mathbb{Z}/n\mathbb{Z})^* = \hat{\mathbb{Z}}^*.$$

But  $\hat{\mathbb{Z}} = \prod_p \mathbb{Z}_p$ , and  $\mathbb{Z}_p^* \cong \mathbb{Z}_p \times \mathbb{Z}/(p-1)\mathbb{Z}$  for  $p \neq 2$  and  $\mathbb{Z}_2^* \cong \mathbb{Z}_2 \times \mathbb{Z}/2\mathbb{Z}$ .

Consequently,  $G(\Omega|\mathbb{Q}) \cong \hat{\mathbb{Z}}^* \cong \hat{\mathbb{Z}} \times \hat{T}$ , where  $\hat{T} = \prod_{p \neq 2} \mathbb{Z}/(p-1)\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ .

This shows that

the torsion subgroup  $T$  of  $G(\Omega|\mathbb{Q})$  is isomorphic to the torsion subgroup of  $\hat{T}$ .

Since the latter contains the group  $\bigoplus_{p \neq 2} \mathbb{Z}/(p-1)\mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z}$ , we see that

the closure  $\bar{T}$  of  $T$  is isomorphic to  $\hat{T}$ . Now, if  $\tilde{\mathbb{Q}}$  is the fixed field of  $T$ , this

implies that  $G(\tilde{\mathbb{Q}}|\mathbb{Q}) = G(\Omega|\mathbb{Q})/\bar{T} \cong \hat{\mathbb{Z}}$ .

That is, ...

Since A, we find B. But C, and D and E.

Consequently, F, where G. This shows that H.

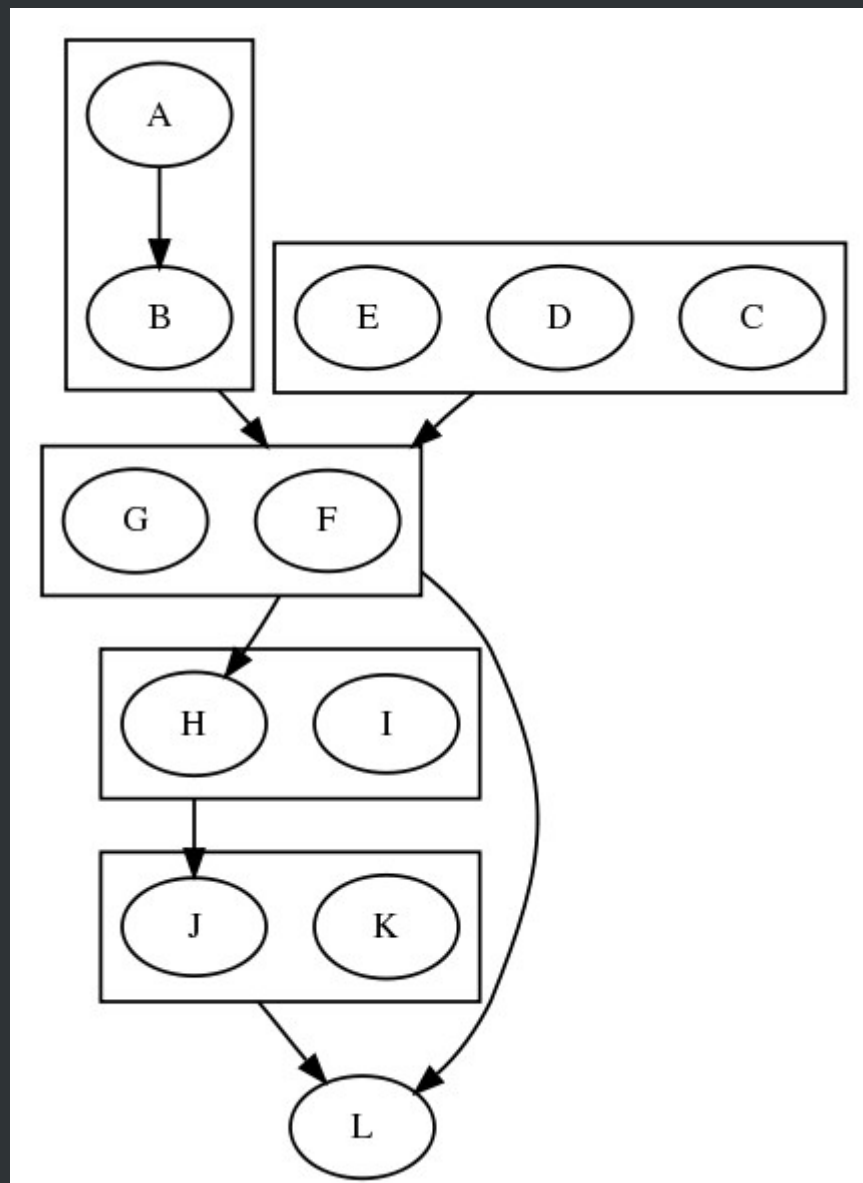
Since I, we see that J. Now, if K, this implies that L.

# Thm lang. vs. Pf lang.

Since A, we find B. But C,  
and D and E.

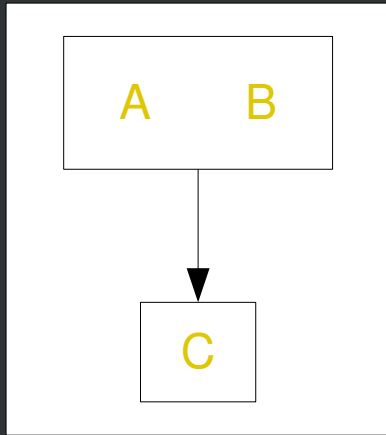
Consequently, F, where G.  
This shows that H.

Since I, we see that J. Now,  
if K, this implies that L.

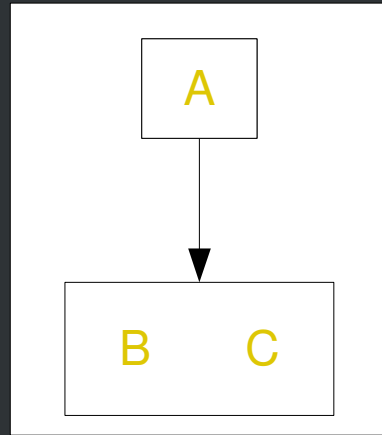


# Propositional chunking

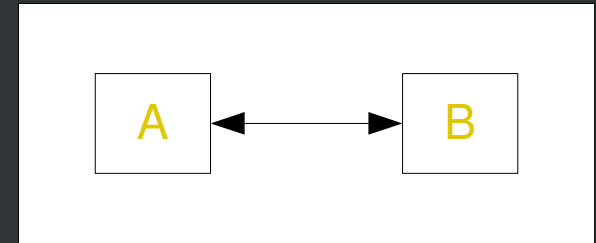
$A \wedge B \rightarrow C$



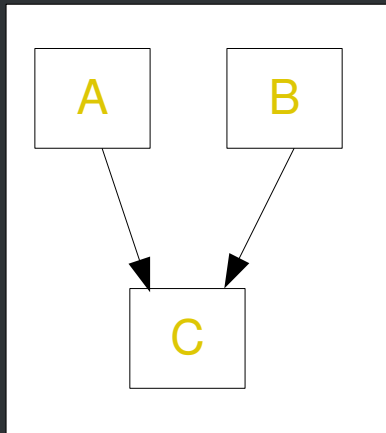
$A \rightarrow B \wedge C$



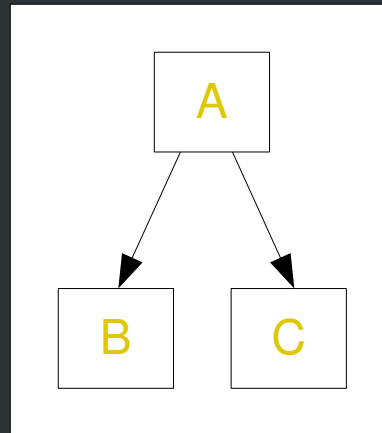
$A \leftrightarrow B$



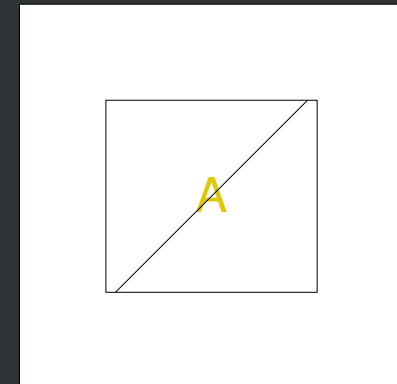
$A \vee B \rightarrow C$



$A \rightarrow B \vee C$  ?



$\neg A$  ?



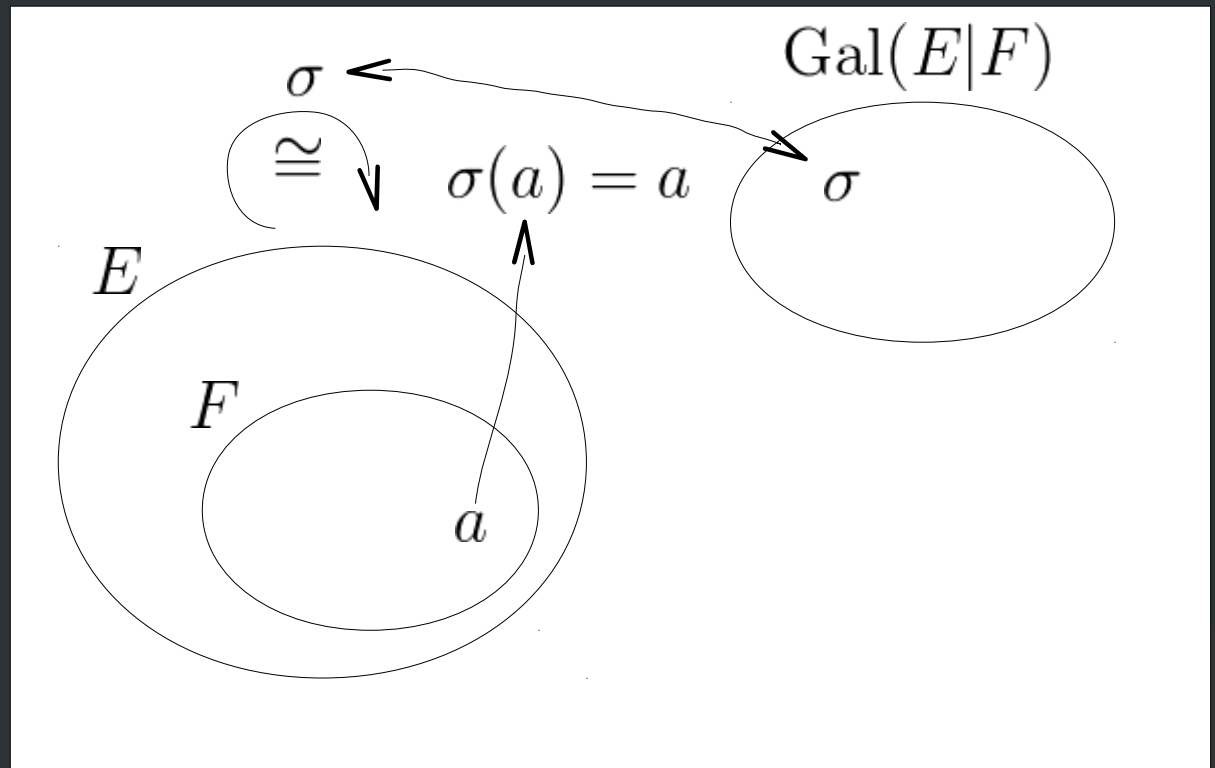
$A \wedge \neg B \rightarrow C$  ?

# NL $\rightarrow$ LDR

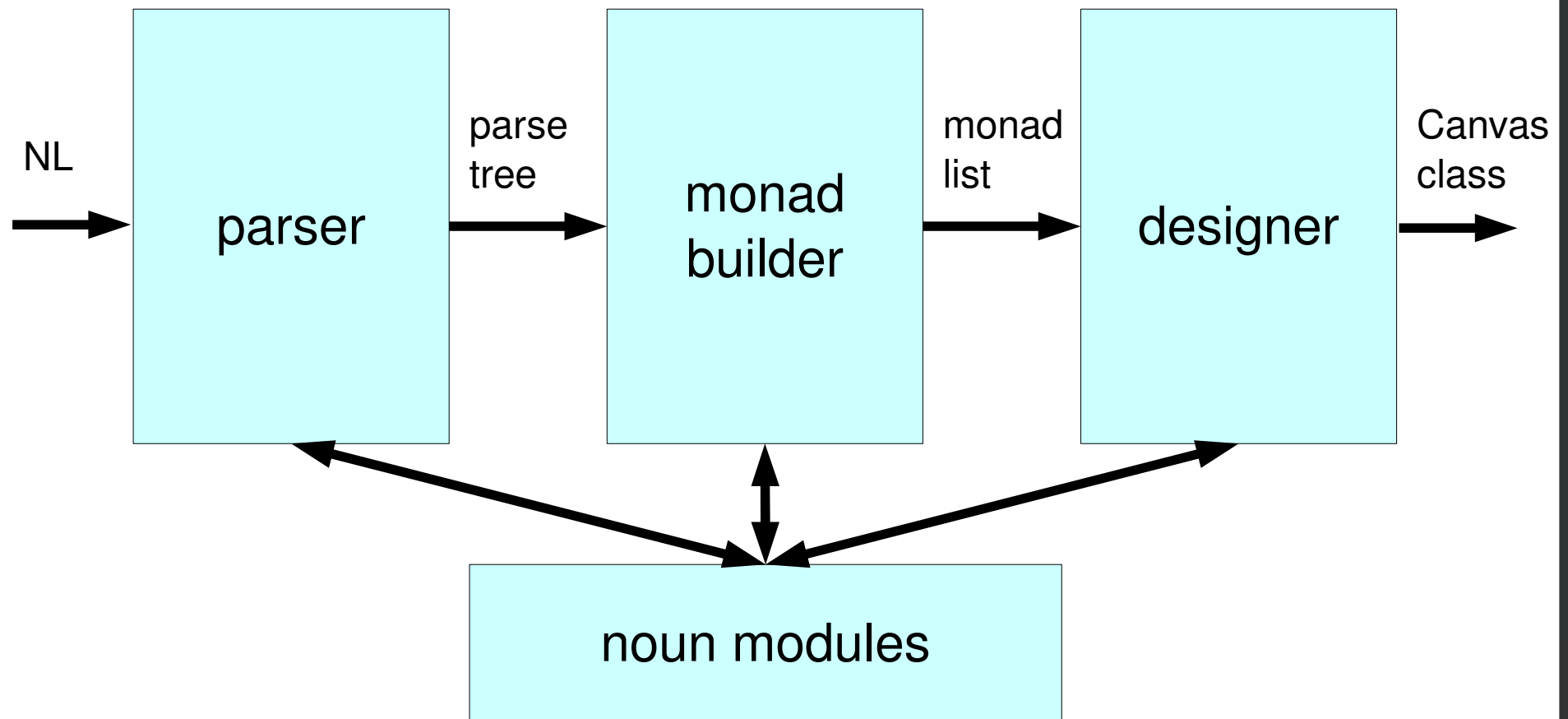
Input:

Let  $E|F$  be fields, and let  $G$  be the Galois group of  $E$  over  $F$ .

Output:  
a logical  
representation  
of a drawing  
like this

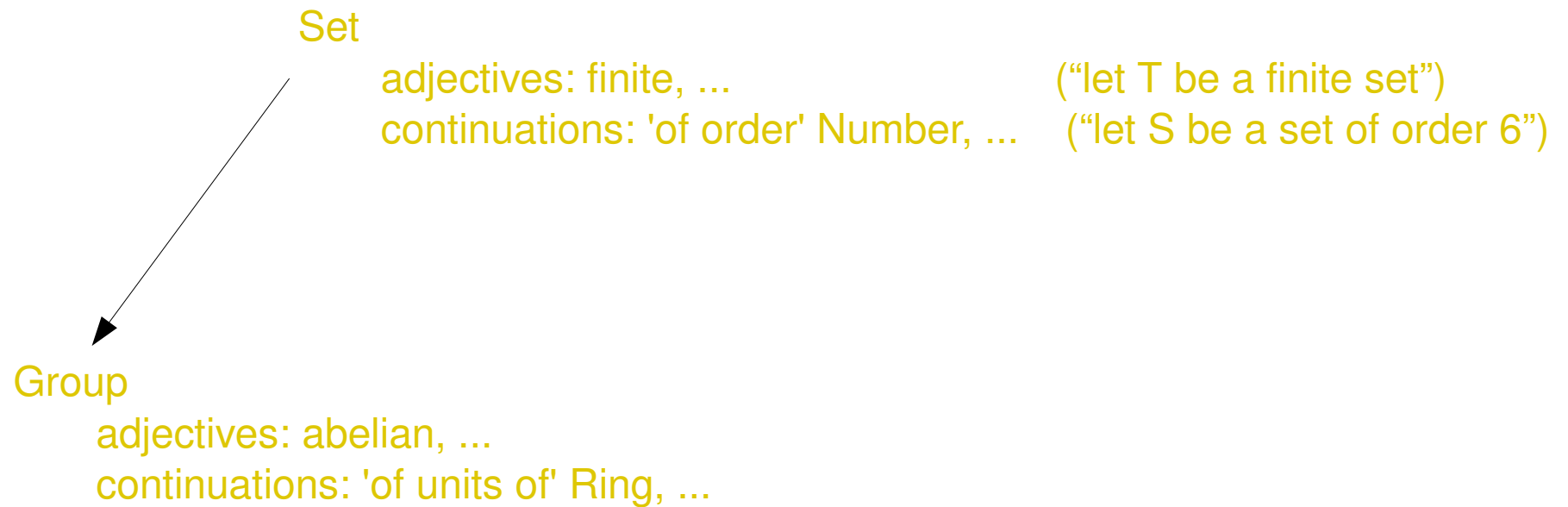


# Architecture

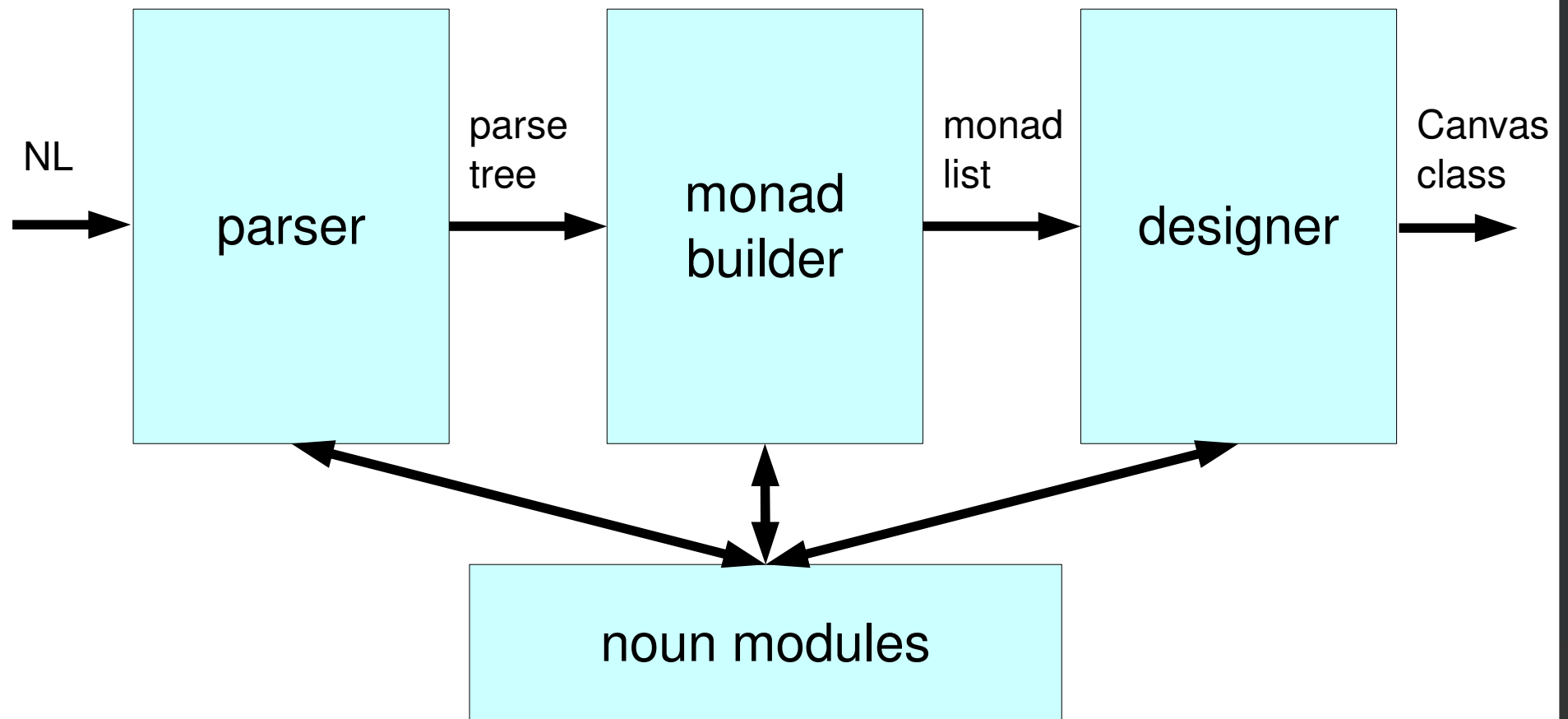


# noun modules

- nouns are classes in a hierarchy



# Architecture



# parser

```
testSens = {
  2:'Let a group  $G$  be given.',
  3:'Let  $G$  be a group.',
  4:'Let  $G$  and  $H$  be groups.',
  5:'Let a group  $G$  and a set  $S$  be given.',
  6:'Let  $G$  be the Galois group of  $E$  over  $F$ .',
  7:'If  $G$  is a cyclic group, then  $G$  is a group.',
  8:'If  $G$  is a cyclic group, then  $G$  is abelian.',
  9:'Suppose  $G_1$  and  $G_2$  are groups.',
  10:'Suppose that  $G_1, \dots, G_n$  are all groups.',
  11:'Assume that  $G$  is abelian.',
  12:'Let  $E|F$  be an extension field.',
  13:'Suppose  $G$  is a group,  $H$  is a subgroup, and  $n$  is its index.',
  14:'Let  $U$  in  $\mathscr{T}$  be given.',
  15:'Let  $g$  and  $h$  in a group  $G$  be given, where  $G$  is Abelian.',
  16:'Let  $g \in G$  be given, where  $G$  is a group.',
  17:'Let fields  $E|F$  be given, and let  $G$  be the Galois group of  $E$ , over  $F$ .',
  18:'Let  $E|F$  be fields, and let  $G$  be the Galois group of  $E$  over  $F$ .'
```



# parser

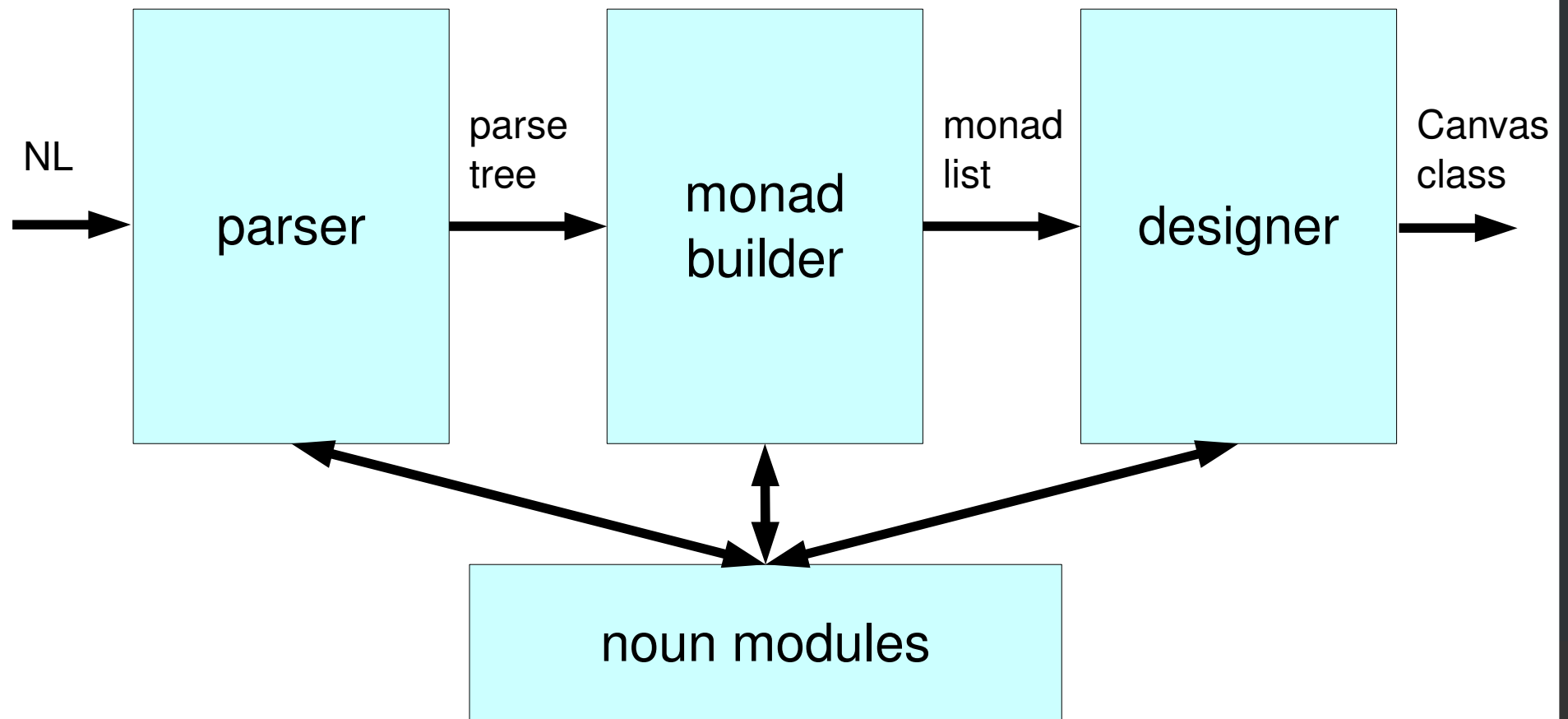
```
SENcore ::= Subject CPL (NP|PN|PA|'given')
          | SymbNPIRNP
          | NP 'consists of' NP

ASSER ::= ( 'for' NP (SubjConj NP)* (COM)? ASSER
          | 'there' ('exists'|'exist'|'is'|'are') NP (SubjConj NP)*
          | SENcore (Conj SENcore)*
          )
          (COM)?
          ( ('where'|'with'|'for'|'such that') ASSER
          | 'and' ASSER
          | 'or' ASSER
          )?
          )
```

# parser

```
SEN ::= ( 'let' SENcore
         | ('assume'|'suppose') ('that')? ASSER
         | 'if' ASSER (COM)? 'then' ASSER
         | 'then' SEN
         | ASSER
       )
      (COM)?
      ( ('where'|'with'|'for'|'such that') ASSER
        | 'and' SEN
        | 'or' SEN
        | 'if and only if' SEN
        | 'implies' ('that')? SEN
      )?
```

# Architecture



# monad builder

Let  $E|F$  be fields, and let  $G$  be the Galois group of  $E$  over  $F$ .

MonadList:

Monad:

Symbol: G

Types: Set([('GaloisGroup.GaloisGroup', ':7:0']])

Relations: Set([])

Monad:

Symbol: E

Types: Set([('Field.ExtensionField', ':1:0']])

Relations: Set([('subset', 2, ('F',)), ':1:0']])

Monad:

Symbol: F

Types: Set([('Field.Field', ':1:0'), ('Set.Set', ':1:0']])

Relations: Set([('subset', 1, ('E',)), ':1:0']])

# “Monads”

- 1714, *Monadologie*
- Every relation into which an object enters is regarded as a property of that object.

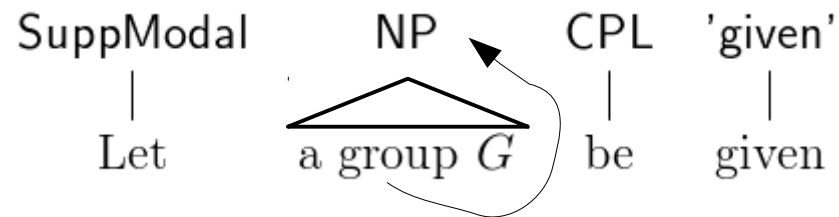


Gottfried Wilhelm von Leibniz

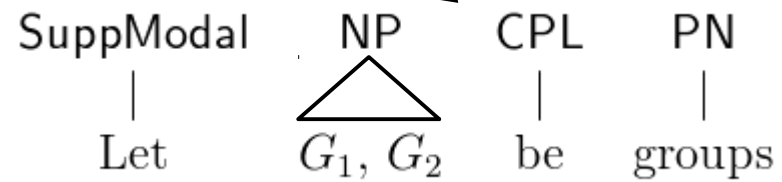
# monad builder algorithm

```
def mainloop(tree):  
    timeStampNPs(tree)  
    ML = MonadList()  
    changed = True  
    while changed:  
        changed = False  
        changed |= applyPNtypes(tree)  
        changed |= applyMathObjectContinTypes(tree)  
        changed |= extractMonads(tree,ML)  
        changed |= typeSpecificMonadExtraction(tree,ML)  
        changed |= monadReInTypes(ML)
```

# noun in NP → type for NP

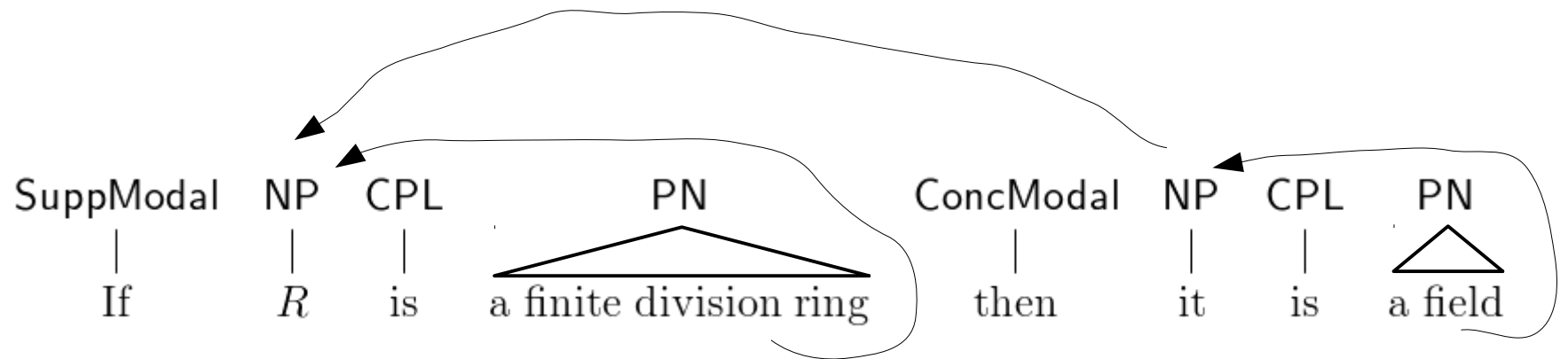


# noun in PN $\rightarrow$ type for NP

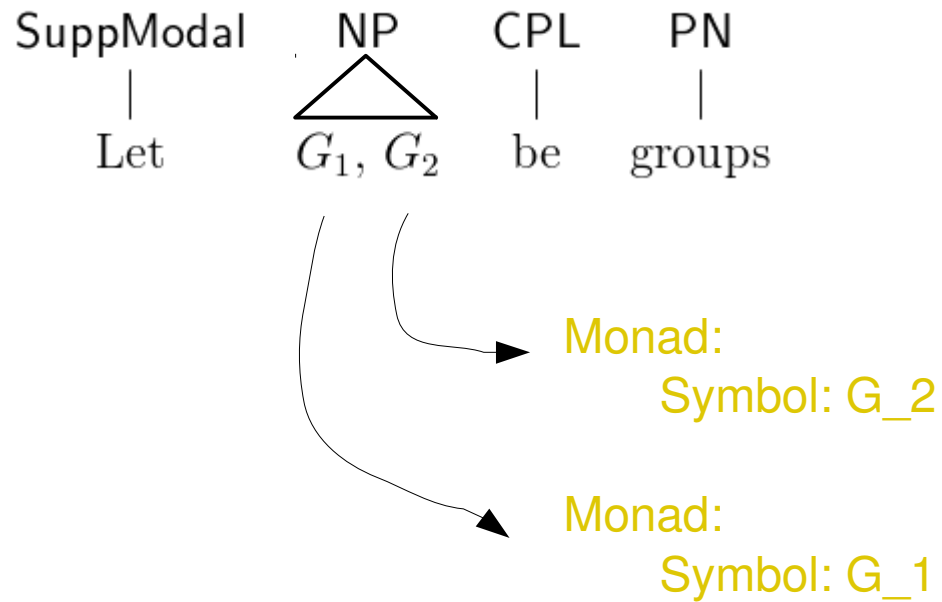




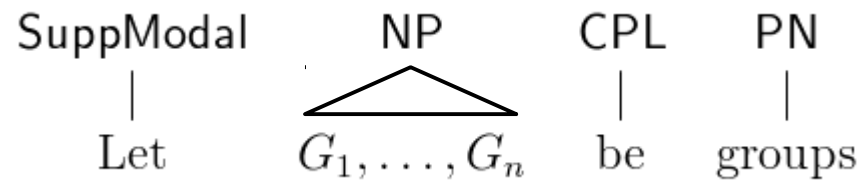
# address of type info is important



# extract symbols

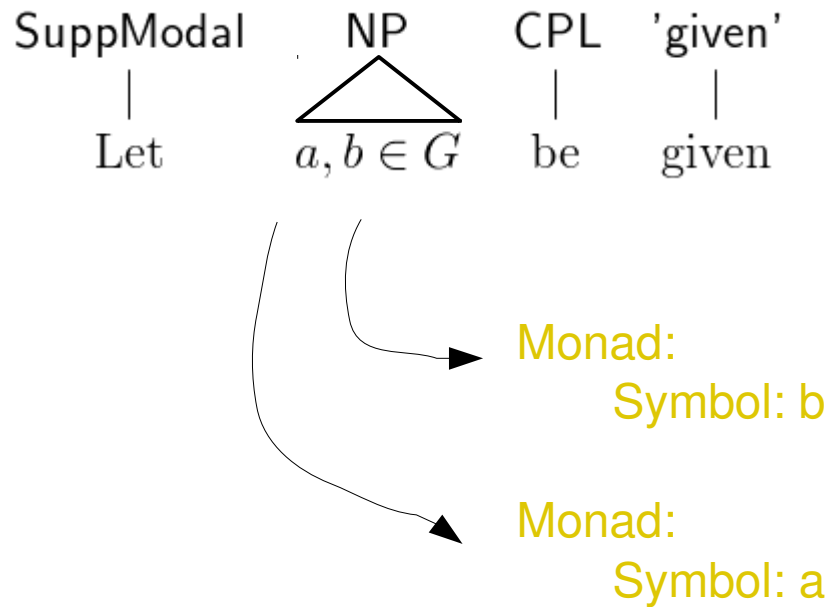


# extract indeterminate list

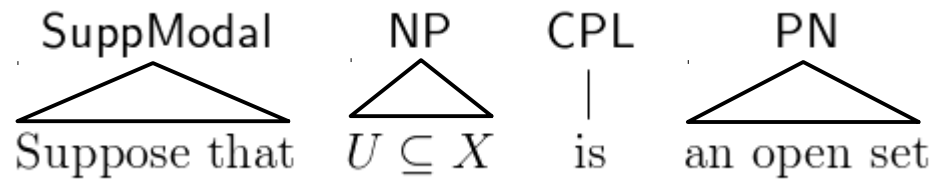


Monad:  
Symbol:  $G_i$

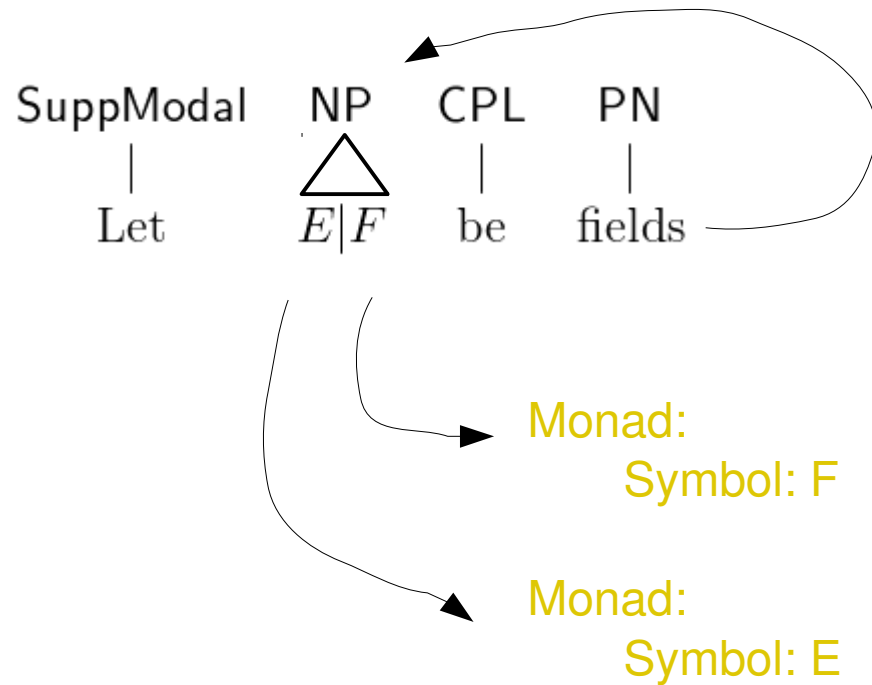
# extract symbols, with “known” IR



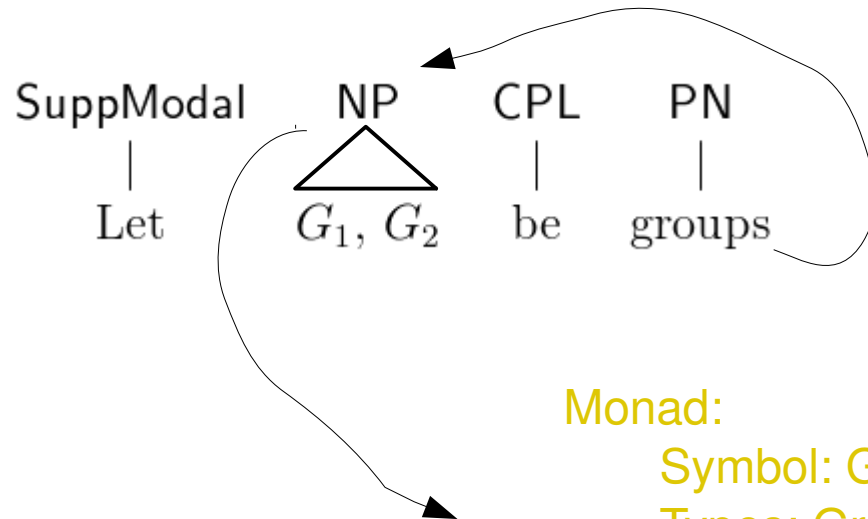
# extract symbols, with “known” IR



# extract symbols, with “unknown” IR



# type from NP → its monads



Monad:

Symbol: G<sub>2</sub>  
Types: Group

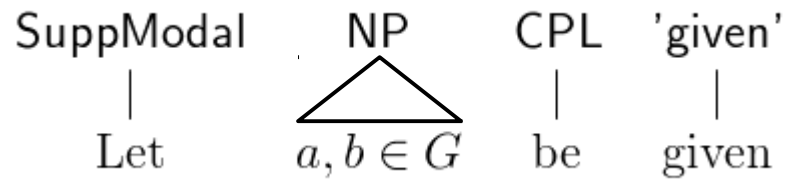
Monad:

Symbol: G<sub>1</sub>  
Types: Group





# add relation to monads



Monad:

Symbol:  $G$

Relations:  $(\text{set\_memb}, 2, a), (\text{set\_memb}, 2, b)$

Monad:

Symbol:  $a$

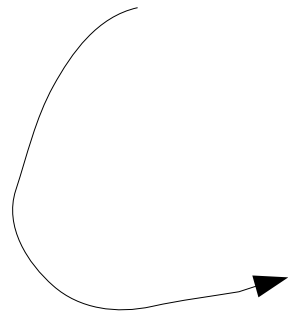
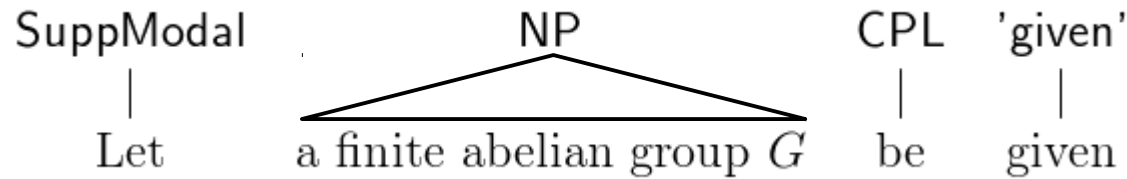
Relations:  $(\text{set\_memb}, 1, G)$

Monad:

Symbol:  $b$

Relations:  $(\text{set\_memb}, 1, G)$

# pass adjectives from NP to monads



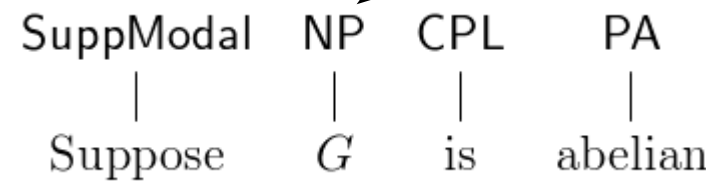
Monad:

Symbol:  $G$

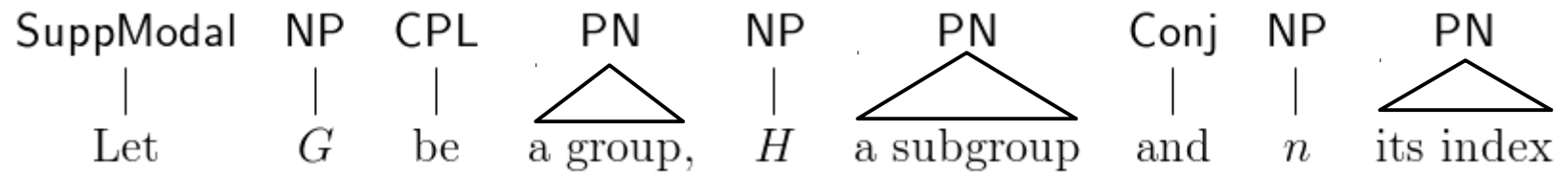
Types: Group

Adjectives: [ (abelian, Group),  
(finite, Set)]

# pass adjective from PA to NP



# Context sensitivity



# Context sensitivity

SuppModal	NP	SubjConj	NP	CPL	PN	SuppModal	NP	CPL
Let	$G_1$	and	$G_2$	be	groups.	Let	$H_1$	be

PN  
a subgroup of  $G_1$

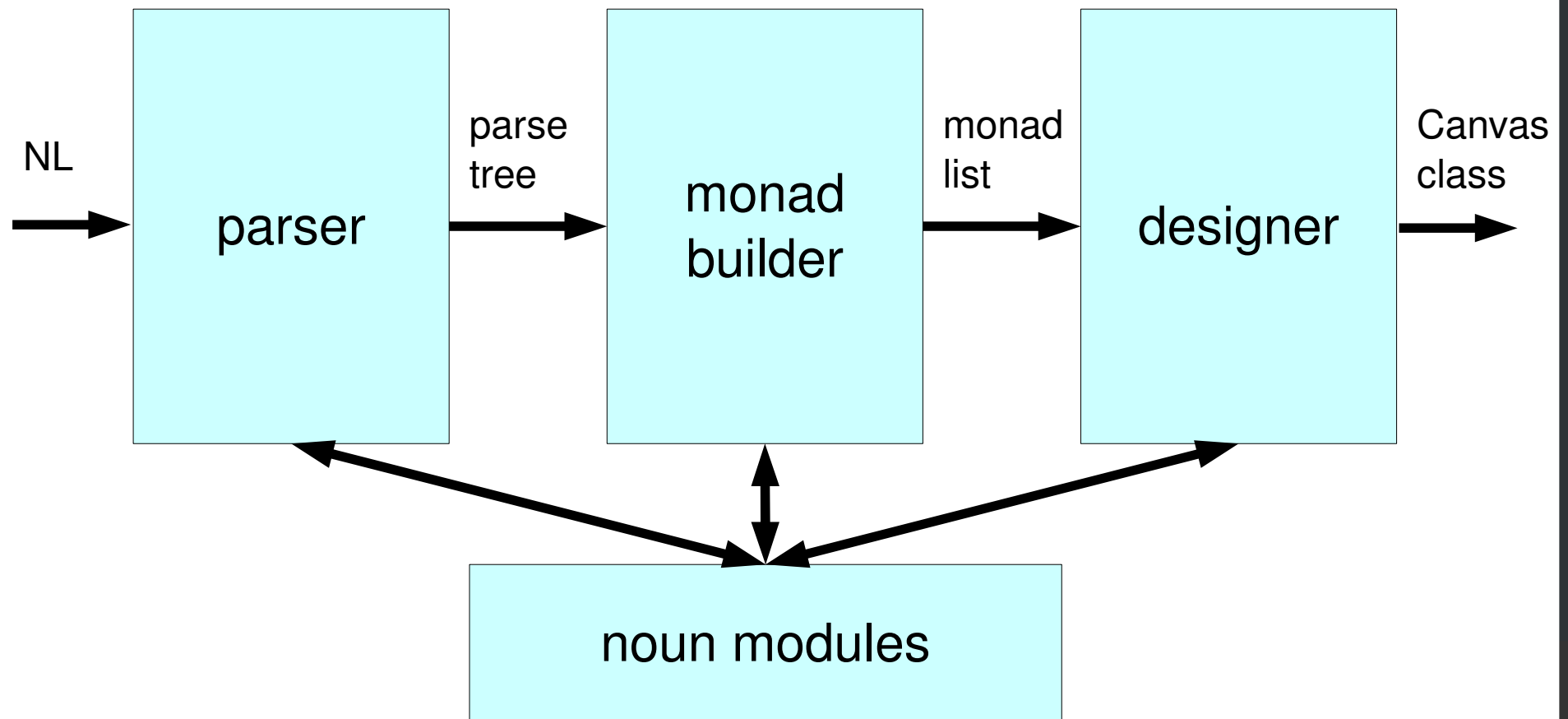
Conj  
and

SuppModal  
let

NP ID  
 $n_1$  be

NP  
the index of  $H_1$

# Architecture

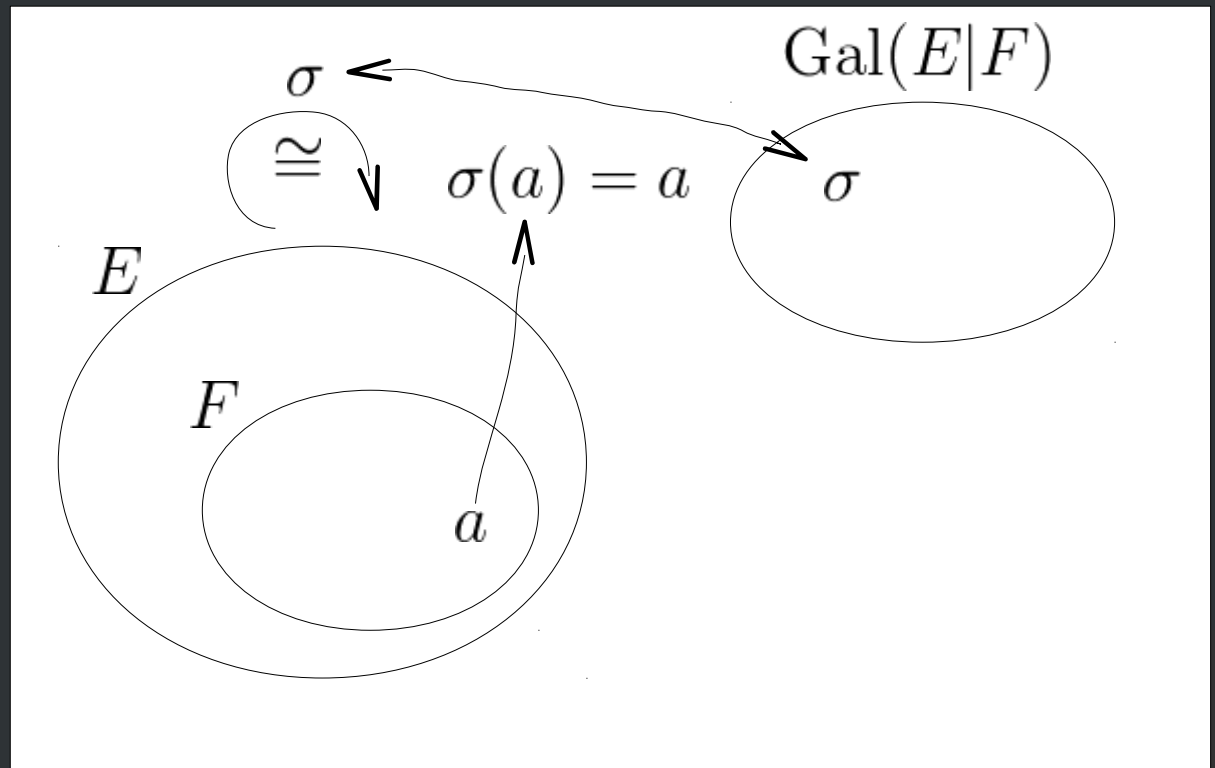


# NL $\rightarrow$ LDR

Input:

Let  $E|F$  be fields, and let  $G$  be the Galois group of  $E$  over  $F$ .

Output:  
a logical  
representation  
of a drawing  
like this



# NL $\rightarrow$ log. rep.

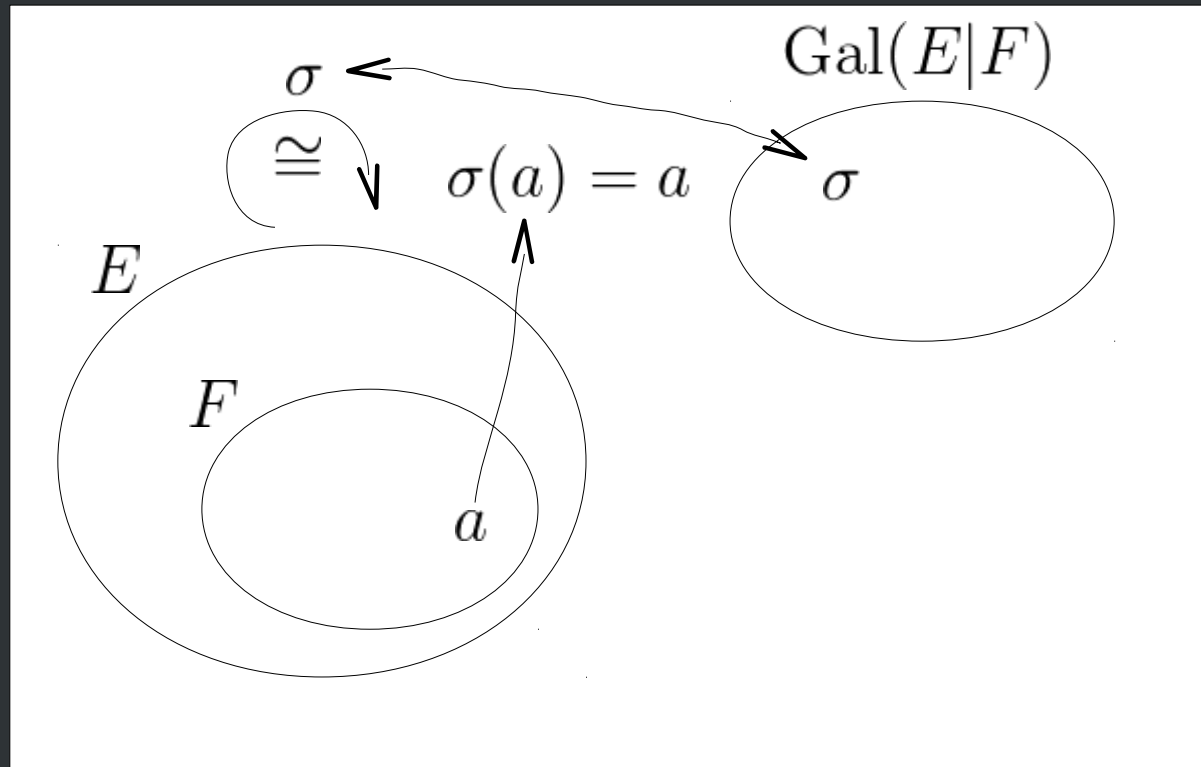
```
def generalPicture():
    E.generalPicture()
    F.generalPicture()

    G = carrier(self)
    G.label('Gal(%s|%s)' % (E,F))
    sigma = G.add_element(
        self.preferred_names
    )

    automorph = E.add(ENDOMAP)
    upperlabel = automorph.label(
        sigma, ABOVE
    )
    automorph.label(r'\cong', BELOW)

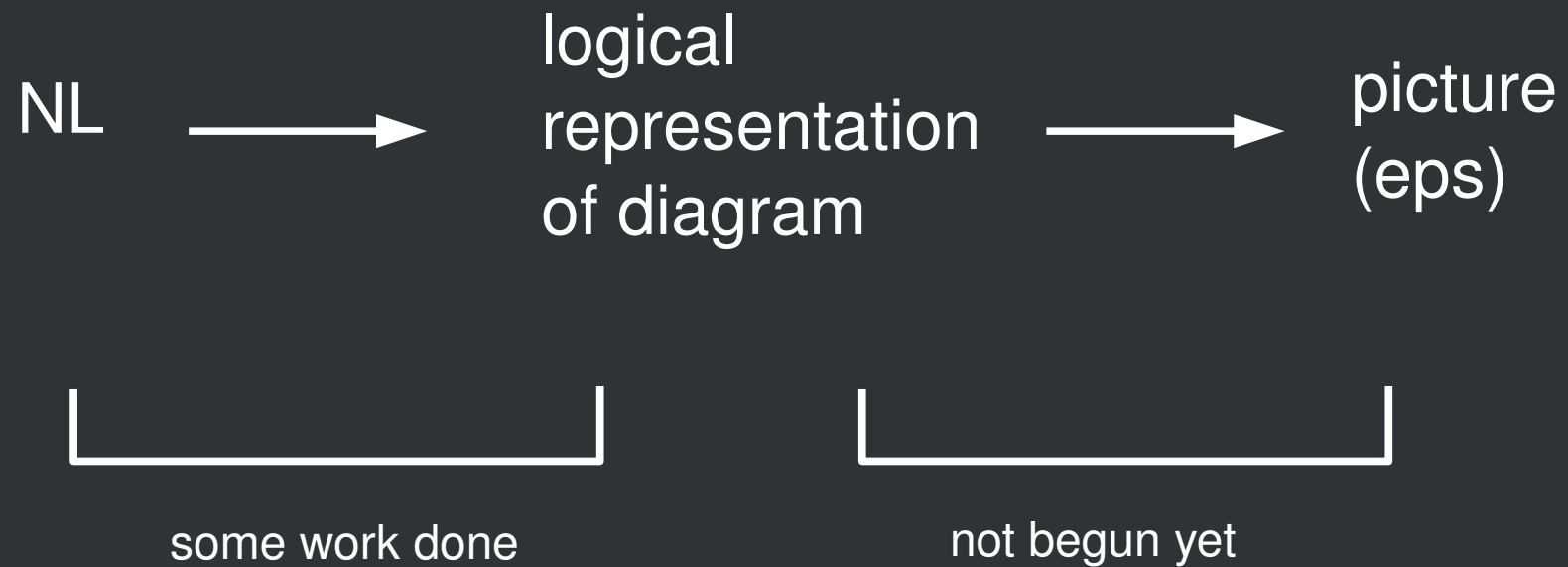
    bidirectional(upperlabel, sigma)

    F.add_callout(
        'a',
        '%s(a) = a' % sigma,
        sigma.NEAR
    )
```





# Modularity



# log. rep. → image

- Pyscript: <http://pyscript.sourceforge.net>

The logo features the word "PyScript" in a black, serif font. A yellow ribbon with a black outline and small yellow dots is draped across the letters, starting from the left and ending at the right, with a small blue star at the tip.

*PyScript*

*Postscript Graphics with Python*