An essential language for declarative business rules

Lex Wedemeijer
Open University in the Netherlands

An essential language for declarative business rules

Lex Wedemeijer
Open University in the Netherlands

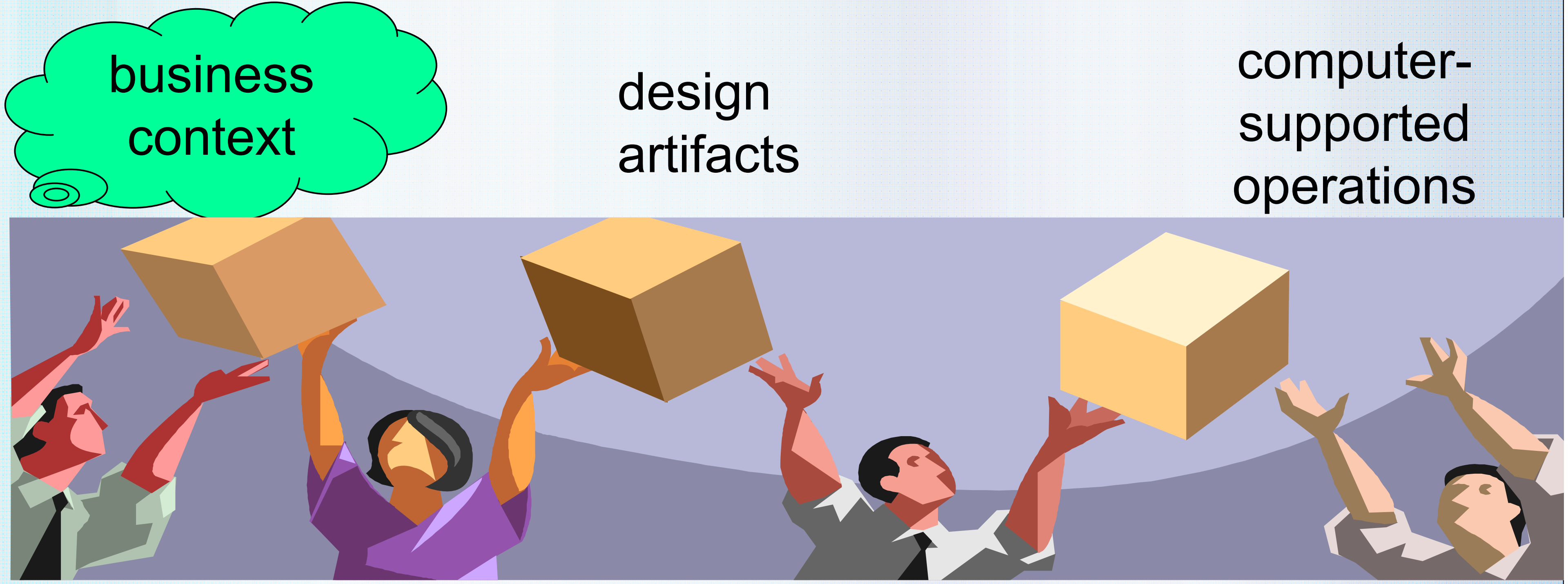
Agenda

- Business Rules
- Rule-engineering approach
- Language specification: steps
- Characteristics
- Discussion

Business Rules

- describe business operations in 'natural' way
- many types of rules
    - ECA-rules, triggers, workflow-rules, derivation rules, transition rules, pre- and post conditions, ...
    - declarative rules: state-oriented, time-invariant
- novice learners

Business Rules



languages:

- natural business jargon
- controlled / semi formal
- exact specifications
- computer lingo

Example: at the IT helpdesk

- natural business jargon:

  every call should get an acceptable response

- controlled language:

  keyword

  concept

  for every call *placed_by* a client there MUST be a response
  that is *available_for* the call which is *accepted_by* the client

  fact-type relation

- exact specs:

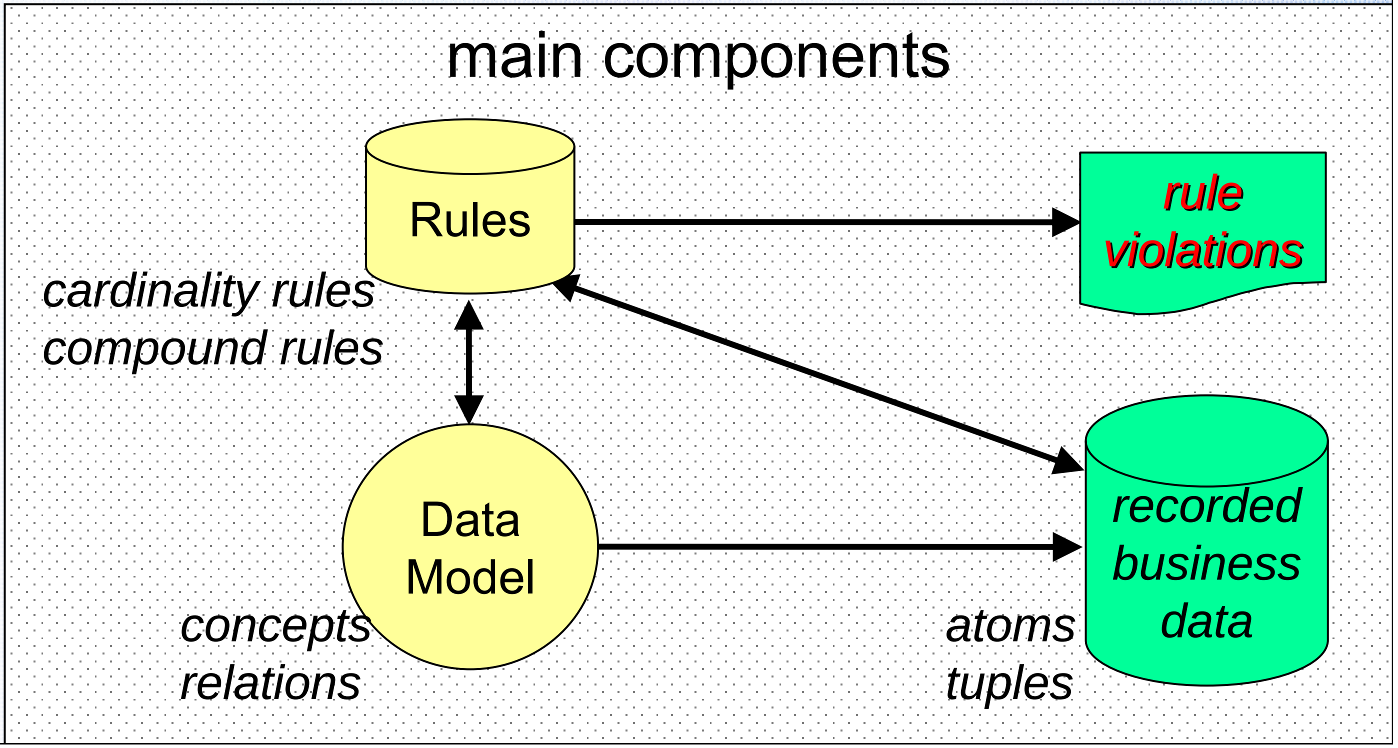
  (Binary Relation Algebra)

- computer lingo:

  (ruleML; SWRL; PRR)

Binary Relation Algebra

- sound math
- suited to declarative rules
- non-computer lingo



Business Rules

business context

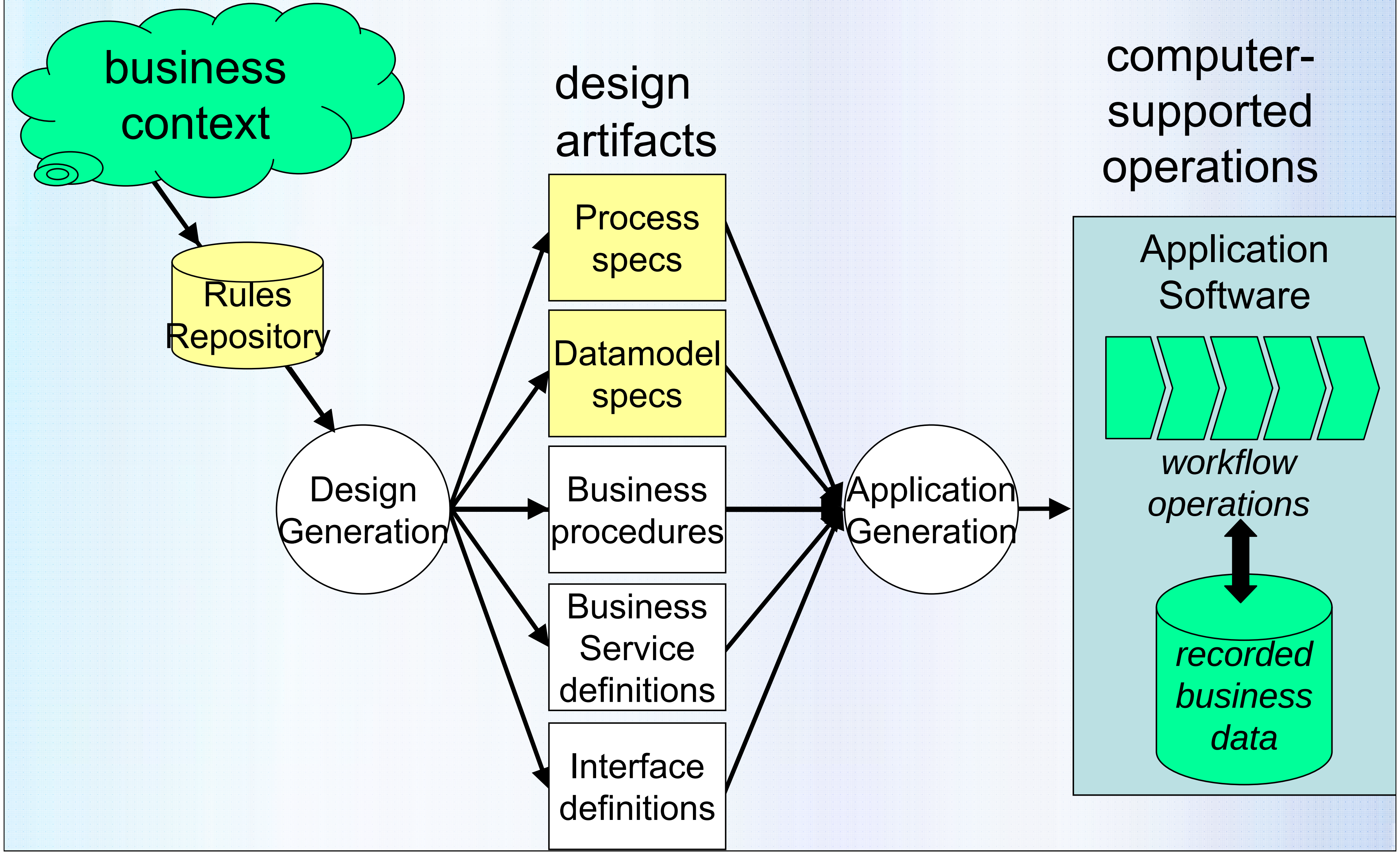
design artifacts

computer-supported operations

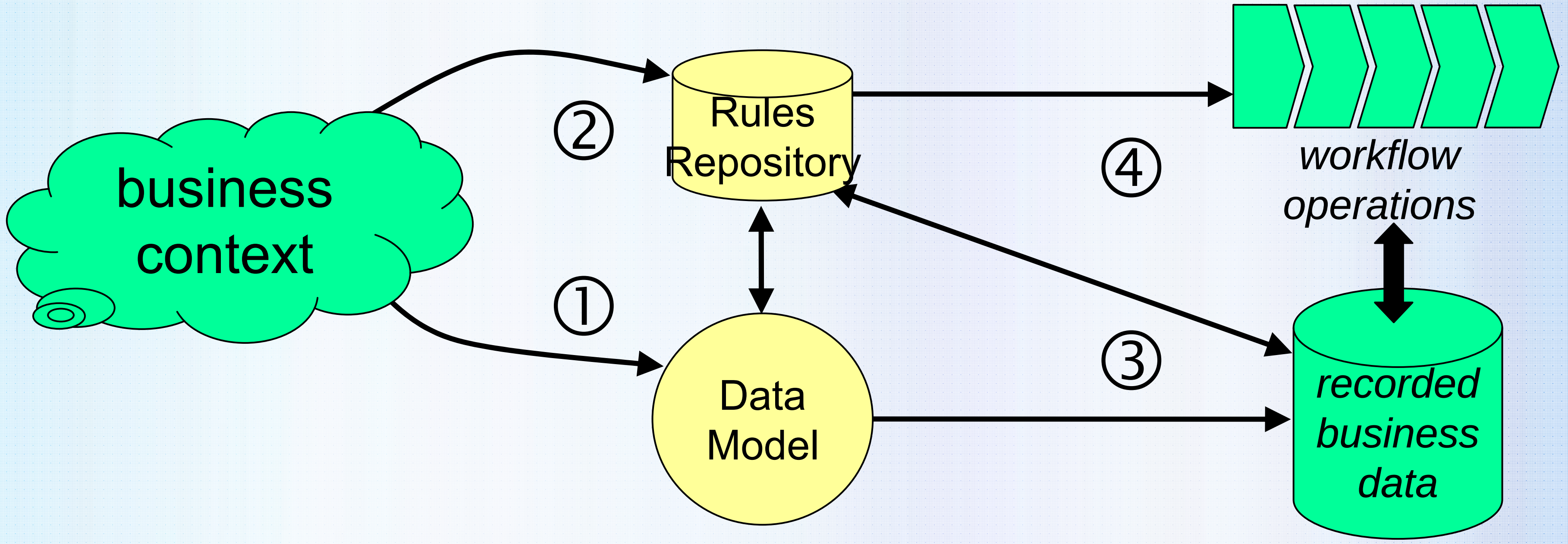
languages:

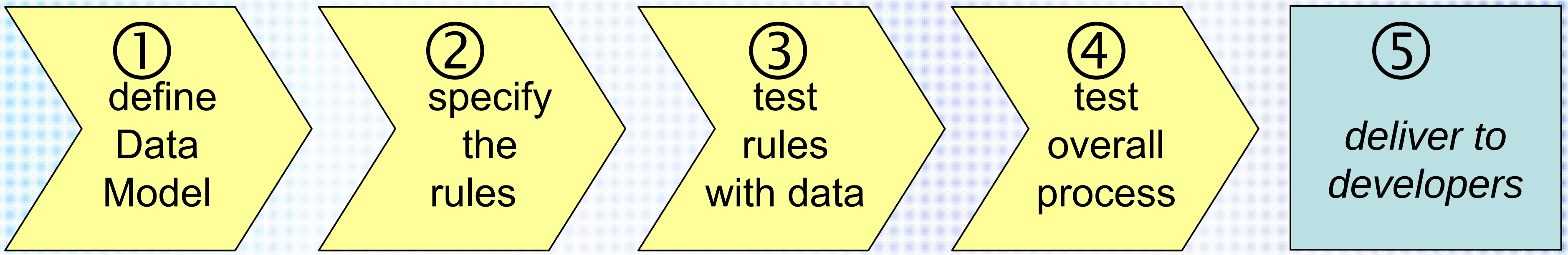
- natural business jargon
- controlled / semi formal
- exact specifications
- computer lingo

Rule-engineering approach



Rule-engineering approach





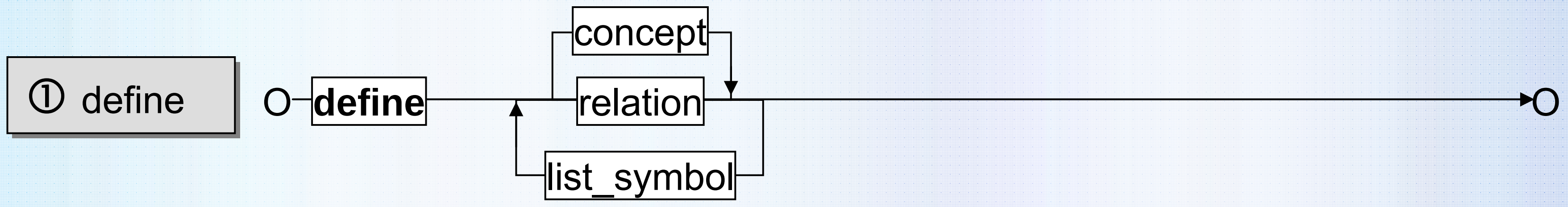
Rule-engineering approach

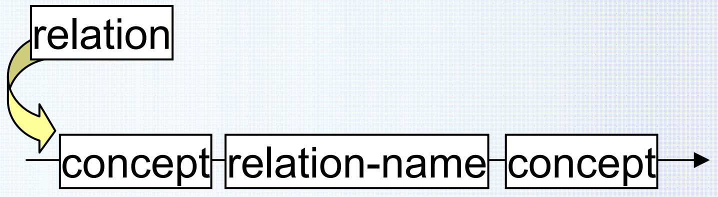
language has 5 statements

- ① define
- ② rule
- ③ populate
- ④ enforce
- ⓪ explain

① define the Model → ② specify the rules → ③ test rules with data → ④ test overall process → ⑤ *deliver to developers*

Specification: step 1





(infix notation)

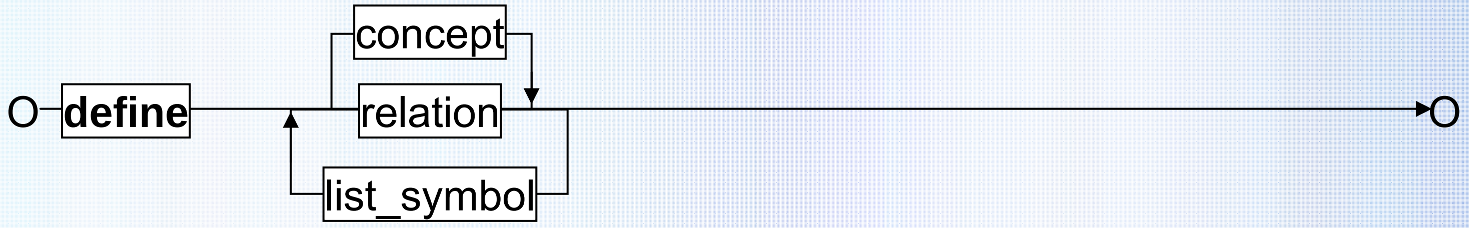
Unconstrained Conceptual Model

Structure =► structural constraints

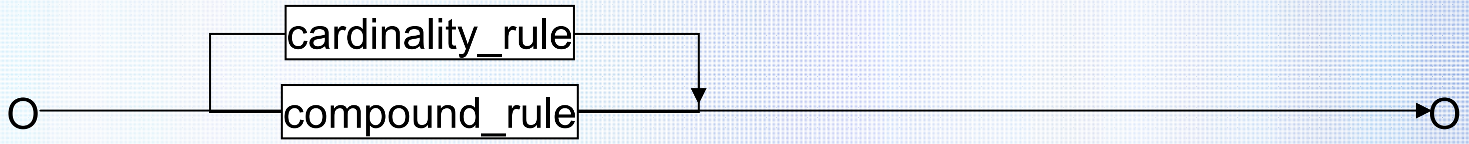
Business Vocabulary

Specification: step 2

① define

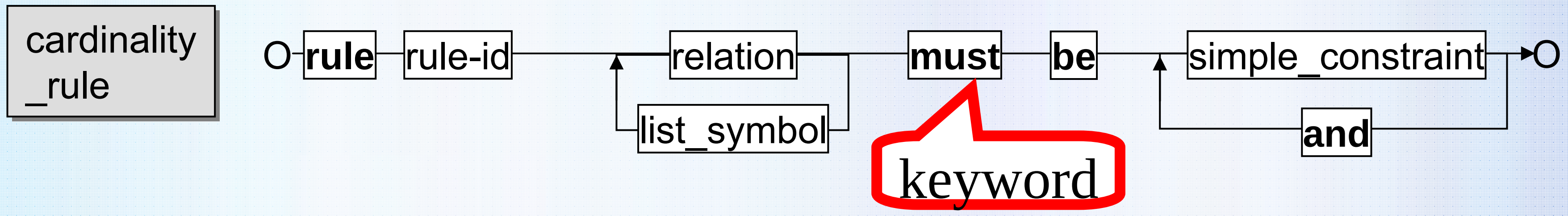


② rule



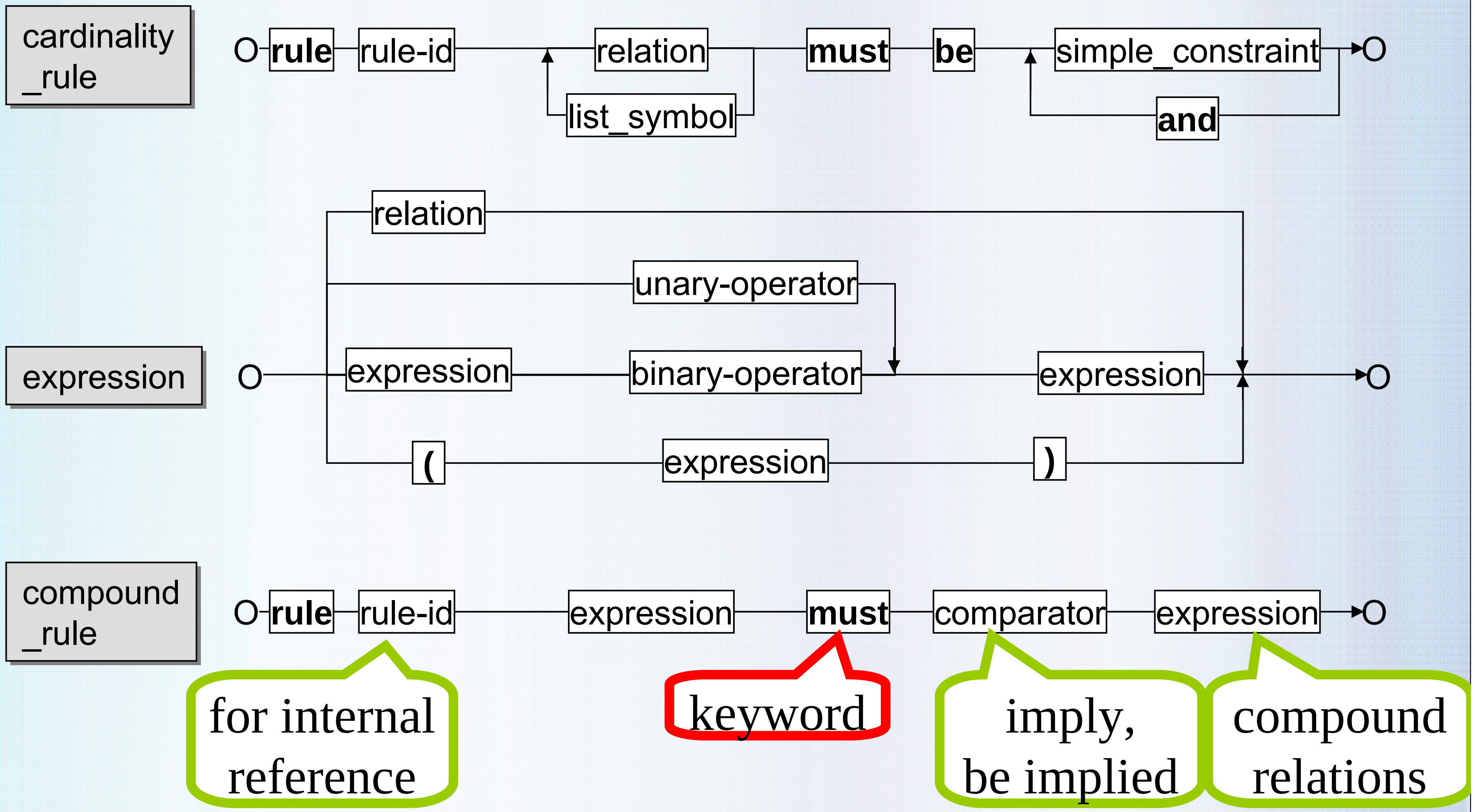
Specification: step 2

detail

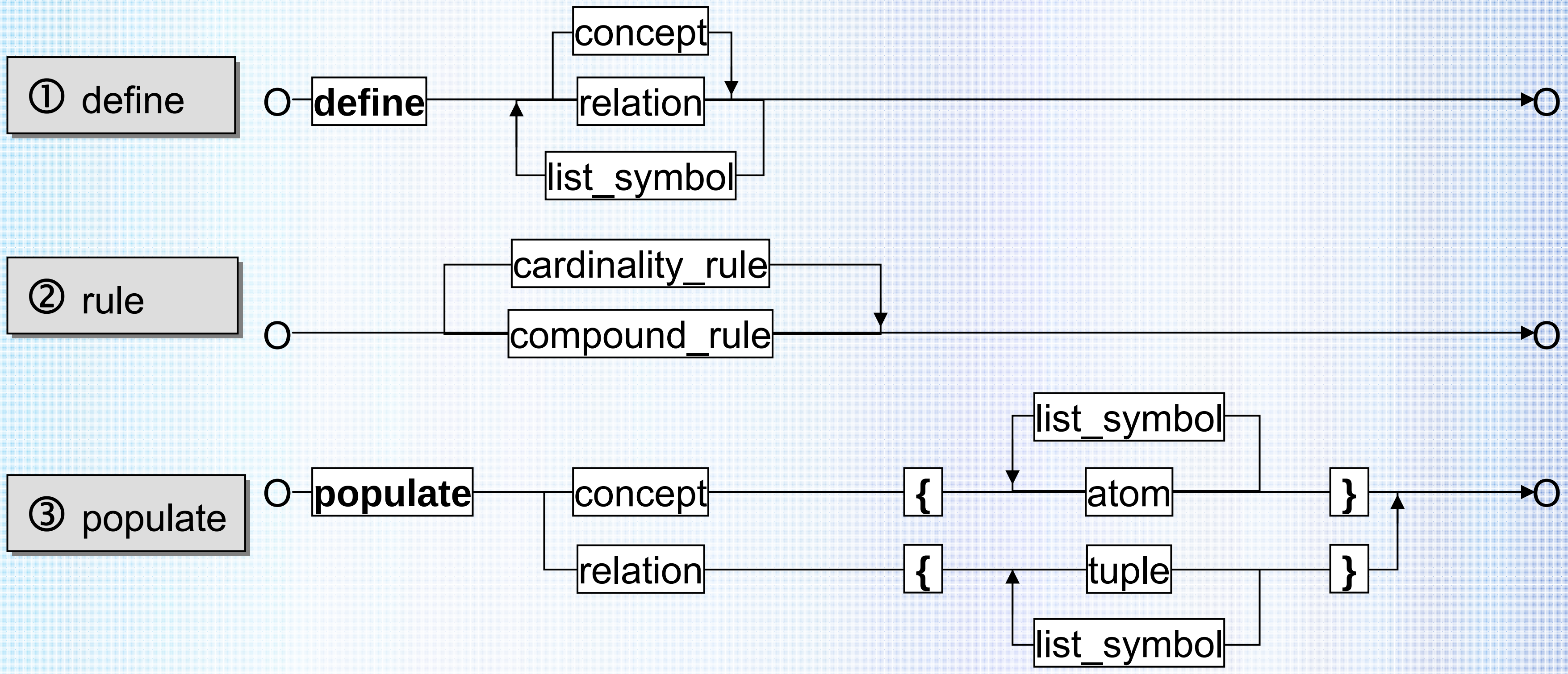


Specification: step 2

detail



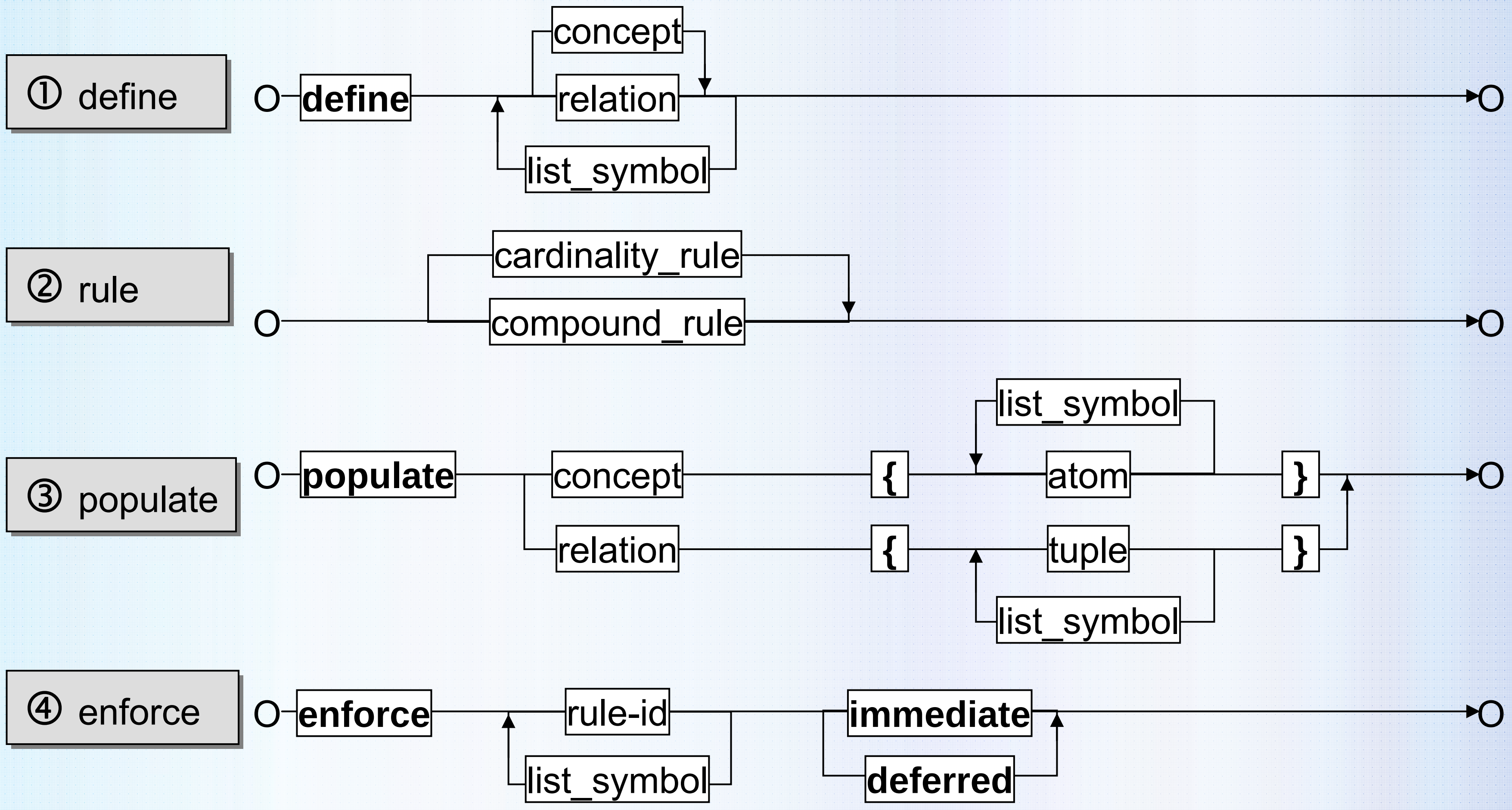
Specification: step 3



Entity integrity, Referential Integrity

Rule violations emerge

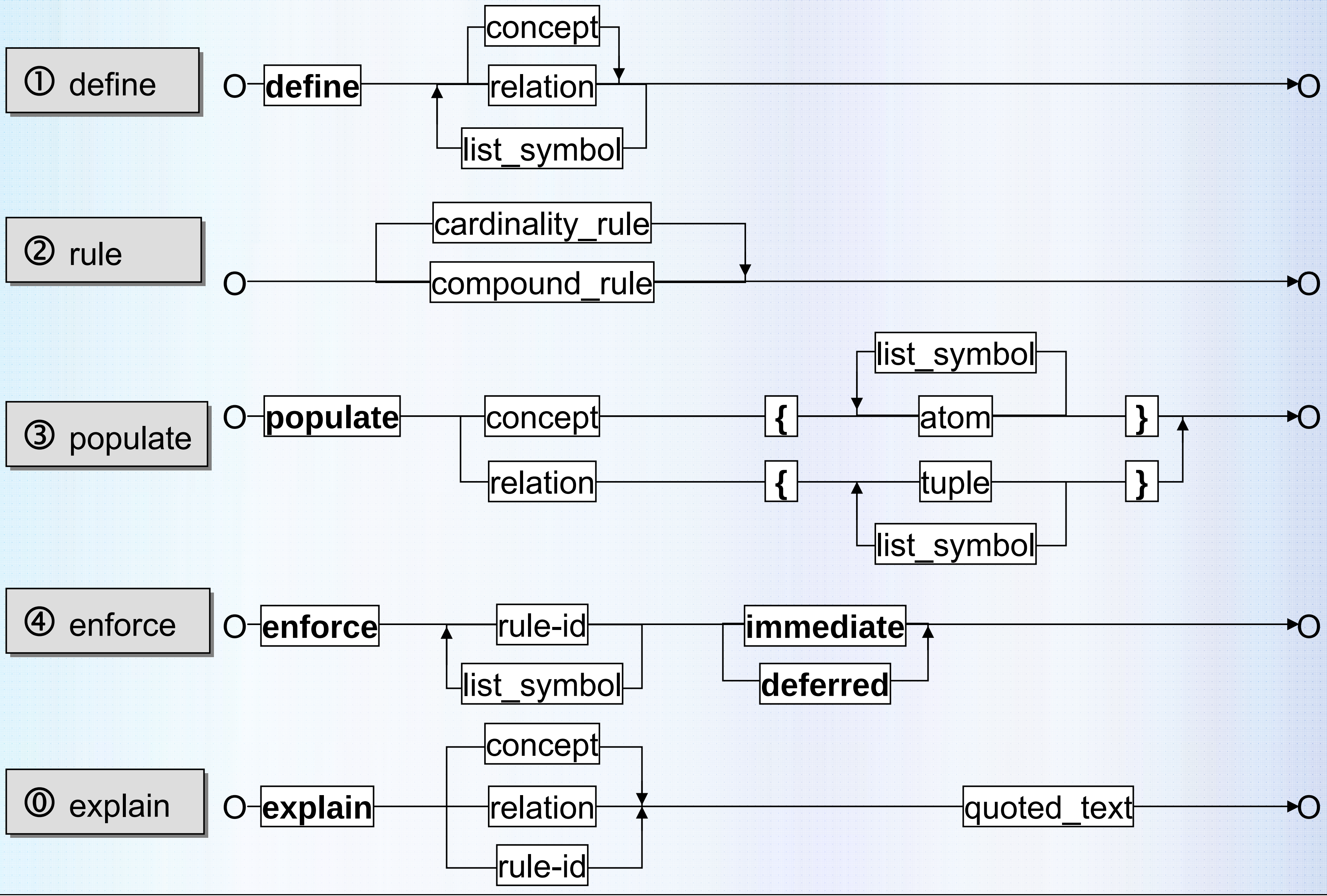
Specification: step 4



Rule violations: prevent or permit

not: how to amend

Specification: any step



Characteristics

1. essential (orthogonal)
2. step-by-step
3. notations kept simple
4. compares to RuleSpeak (almost understandable)

Example: at the IT helpdesk

```
define [call] placed_by [client],
     [response] accepted_by [client],
     [response] available_for [call]

rule 1_cardinal [call] placed_by [client] must be univalent and total

rule 4_helpdesk [call] placed_by [client] must imply
     [call] available_for~ [response];[response] accepted_by [client]

populate [call] placed_by [client] { thursday#1 * lex , fri#2 * kim }

populate [response] available_for [call] { reply#77 * thursday#1 }

enforce 1_cardinal immediate

enforce 4_helpdesk deferred

explain 4_helpdesk "IF a call is placed_by a client
     THEN must a response be available for that call
     AND that response must be accepted_by the client"
```

Characteristics: first findings

- 9 novice students, no prior experience in rule engineering
- compare "IT helpdesk" scripts in

 rich language (Ampersand, see www.tarski.nl)

  versus

 essential language

First findings

rich language — *versus* — essential language

is more suitable for new rule designers	X		X	XXX XXX	X
may be easier to learn for new rule designers	X			XXXX XXXX	
is more educational (learning while designing)	XX	XX		XXX XX	
is easier to use when creating a new rule-based design	XX	X	X	XX XX	X
is better aligned to stepwise design approach	XX	X	XX	XX XX	
may cause more confusion		XXX	XXX	X	XX
is easier to explain to co-workers	XX	XX	X	XX	XX
is simpler to check for errors	X	XXX XX		XXX	
is better in avoiding conflicting (inconsistent) statements	XX	XXX	XX	X	X
has more brief and powerful statements	XX	XX XX	X	X	X

Agenda

- Business Rules
- Rule-engineering approach
- Language specification: steps
- Characteristics
- Discussion

Thank you

IF every <u>question</u> *has got an* <u>acceptable-answer</u>

THEN the <u>speaker</u> *thanks you for* your <u>attention</u>