Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2018, gehouden op dinsdag 6 en woensdag 7 maart 2018 jl. en georganiseerd door CVI i.s.m. NHL/Stenden). Bij elkaar bijna 1450 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Er is nog geen datum bekend voor een volgend NIOC na het niet doorgaan van NIOC 2020 i.v.m. COVID-19. Het NIOC bestuur beraadt zich over een mogelijk vervolg.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

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

business context

design artifacts

computer-supported operations

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:

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

  keyword

  concept

  fact-type relation

- exact specs:

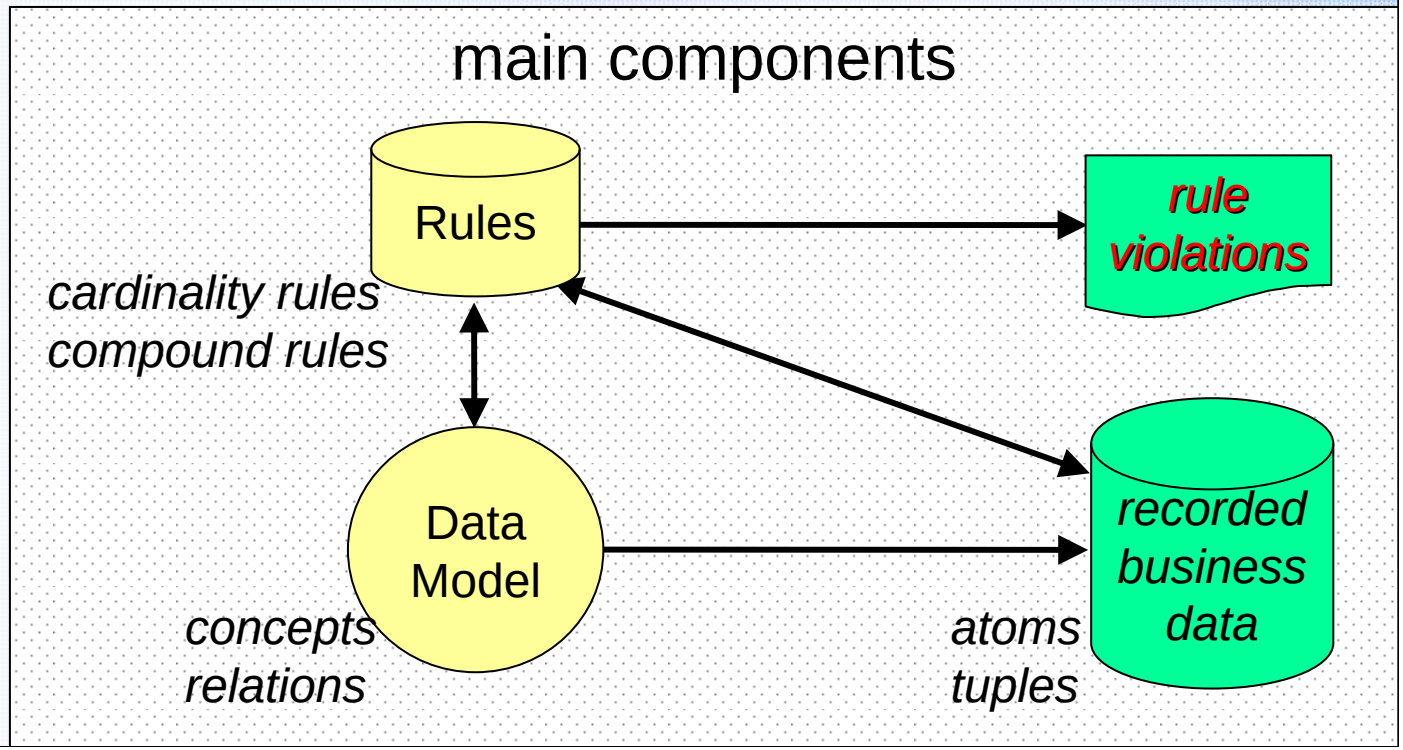
  (Binary Relation Algebra)

- computer lingo:

  (ruleML; SWRL; PRR)

Binary Relation Algebra

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



Business Rules

business context

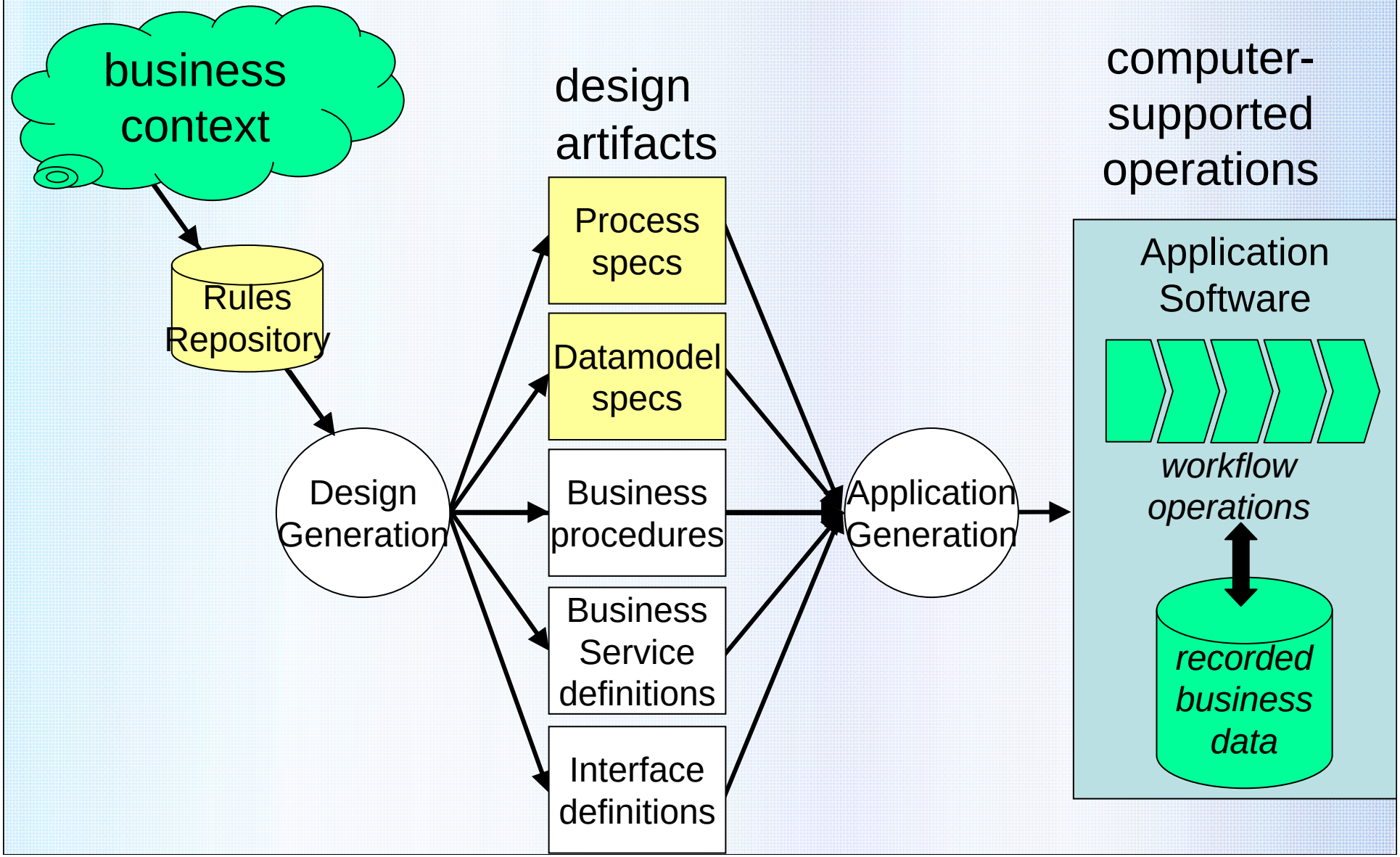
design artifacts

computer-supported operations

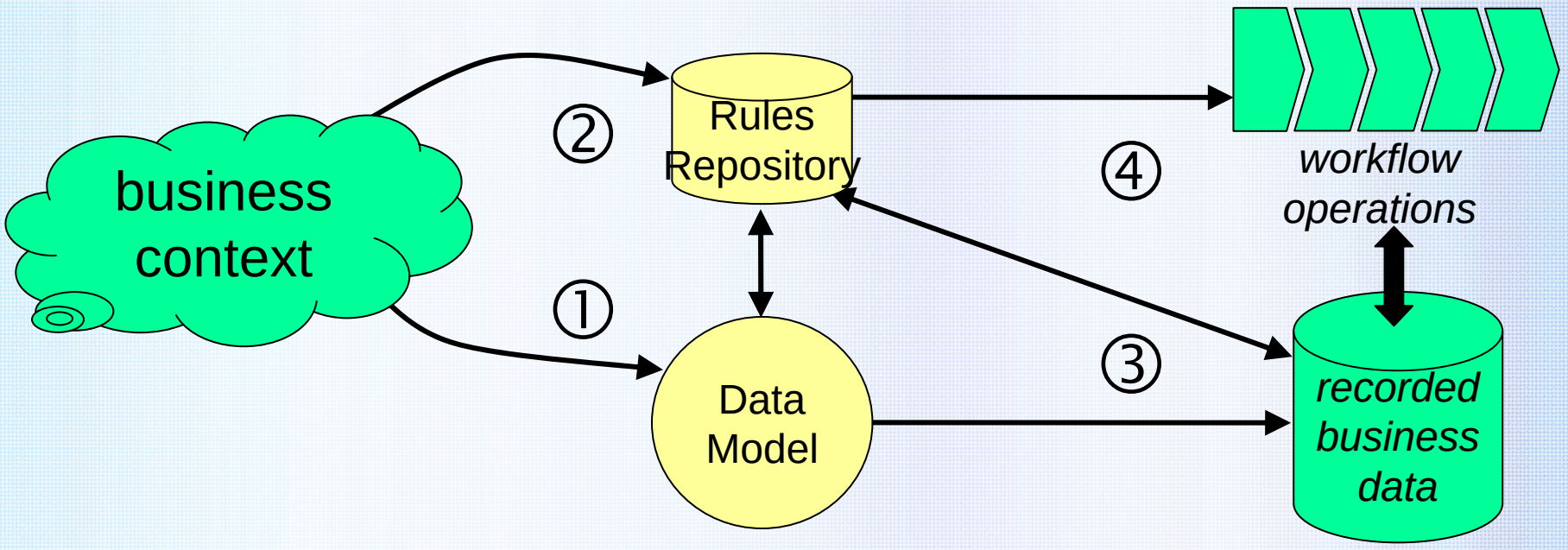
languages:

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

Rule-engineering approach



Rule-engineering approach



① define Data Model

② specify the rules

③ test rules with data

④ test overall process

⑤ *deliver to developers*

Rule-engineering approach

language has 5 statements

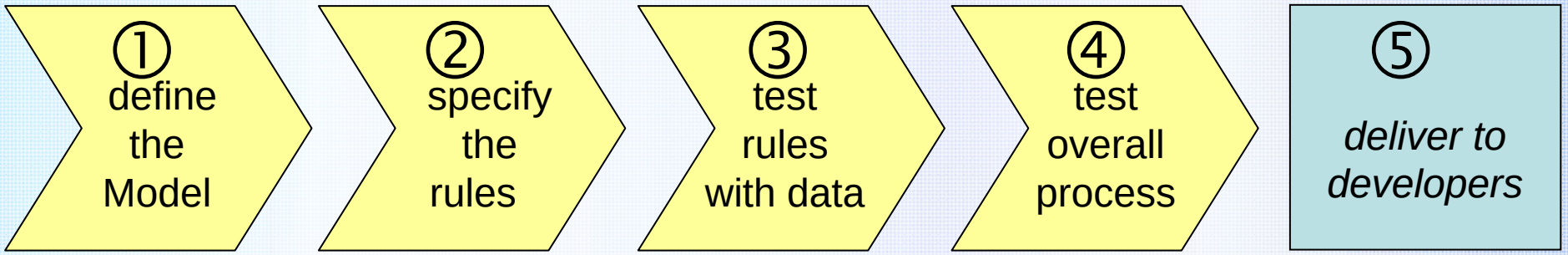
① define

② rule

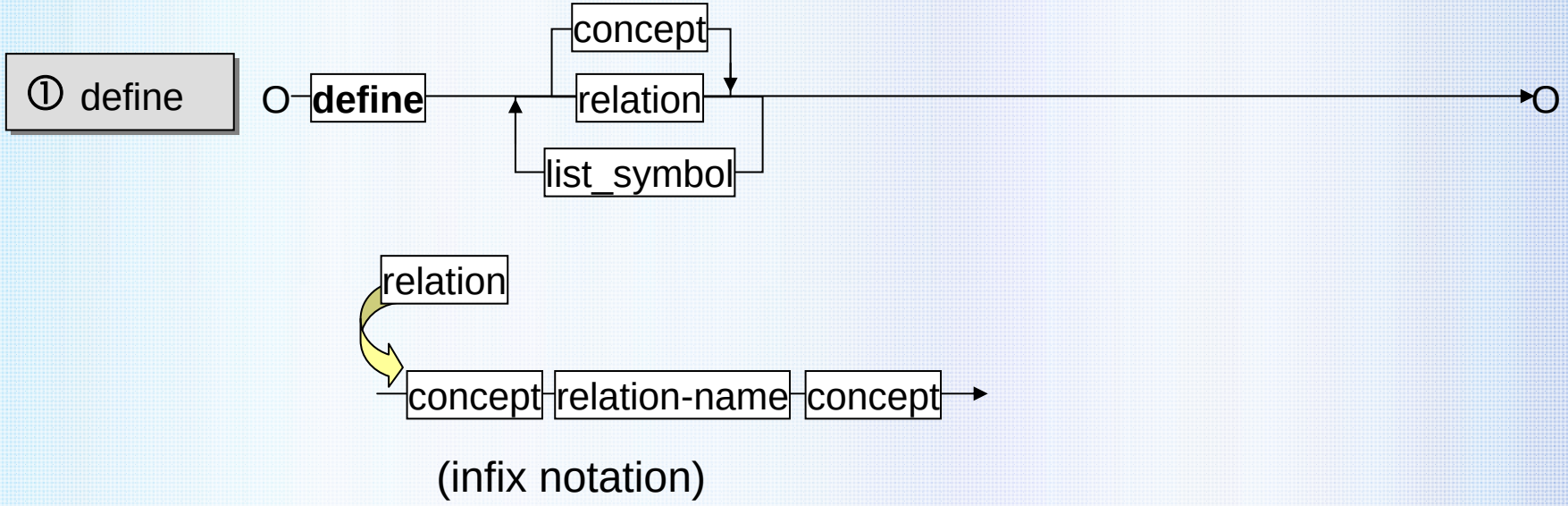
③ populate

④ enforce

⓪ explain



Specification: step 1



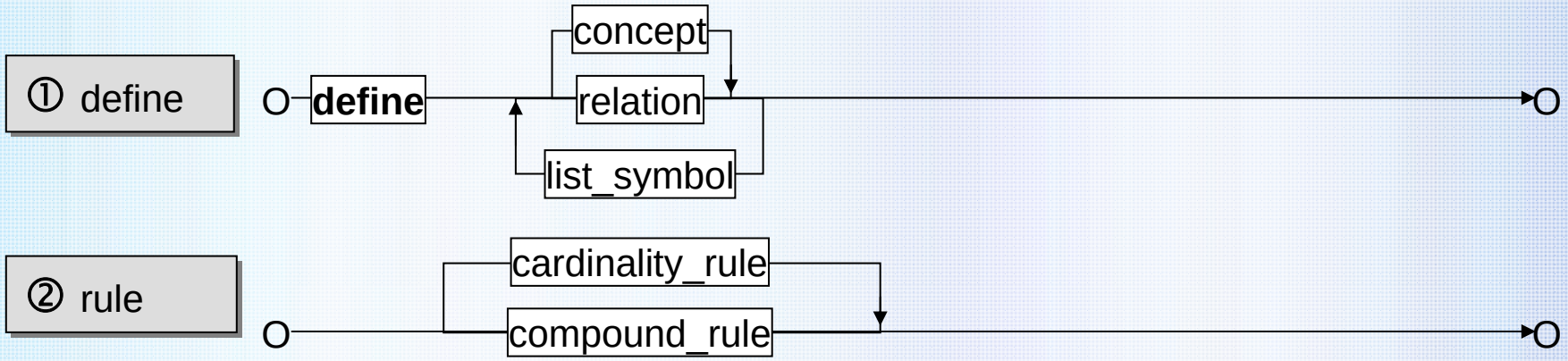
(infix notation)

Unconstrained Conceptual Model

Structure =► structural constraints

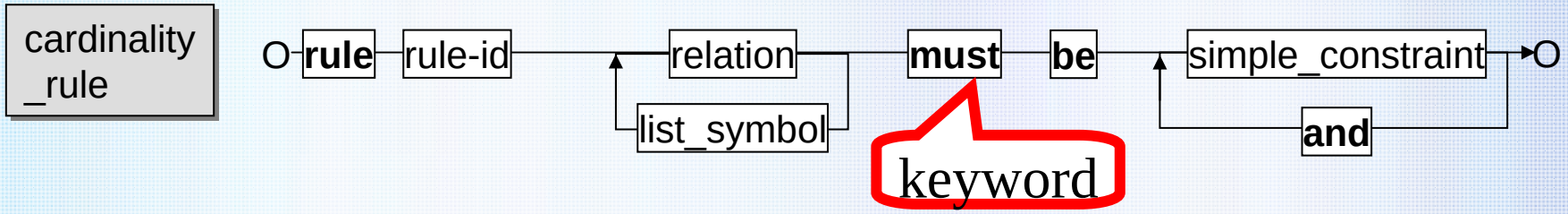
Business Vocabulary

Specification: step 2



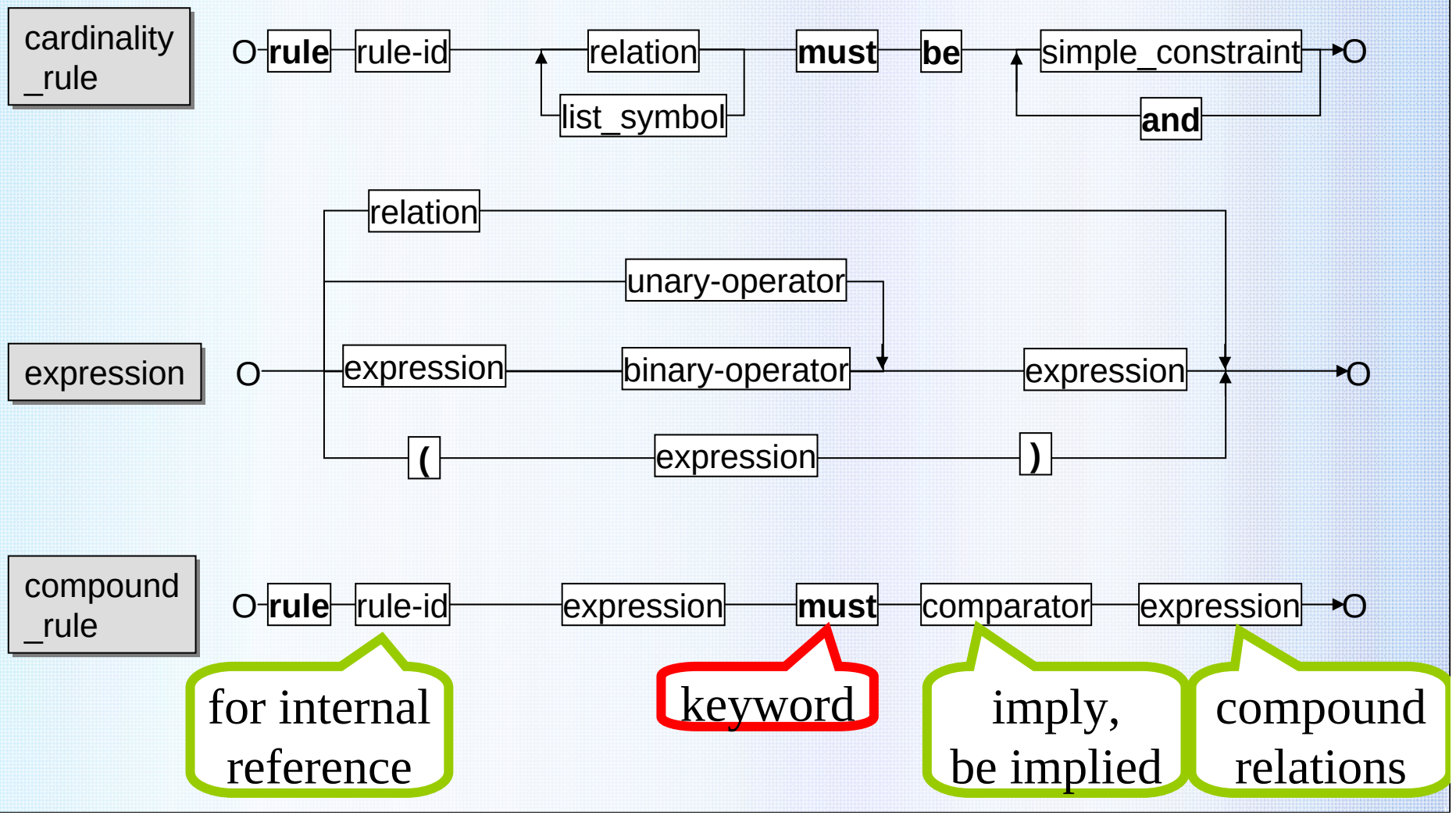
Specification: step 2

detail

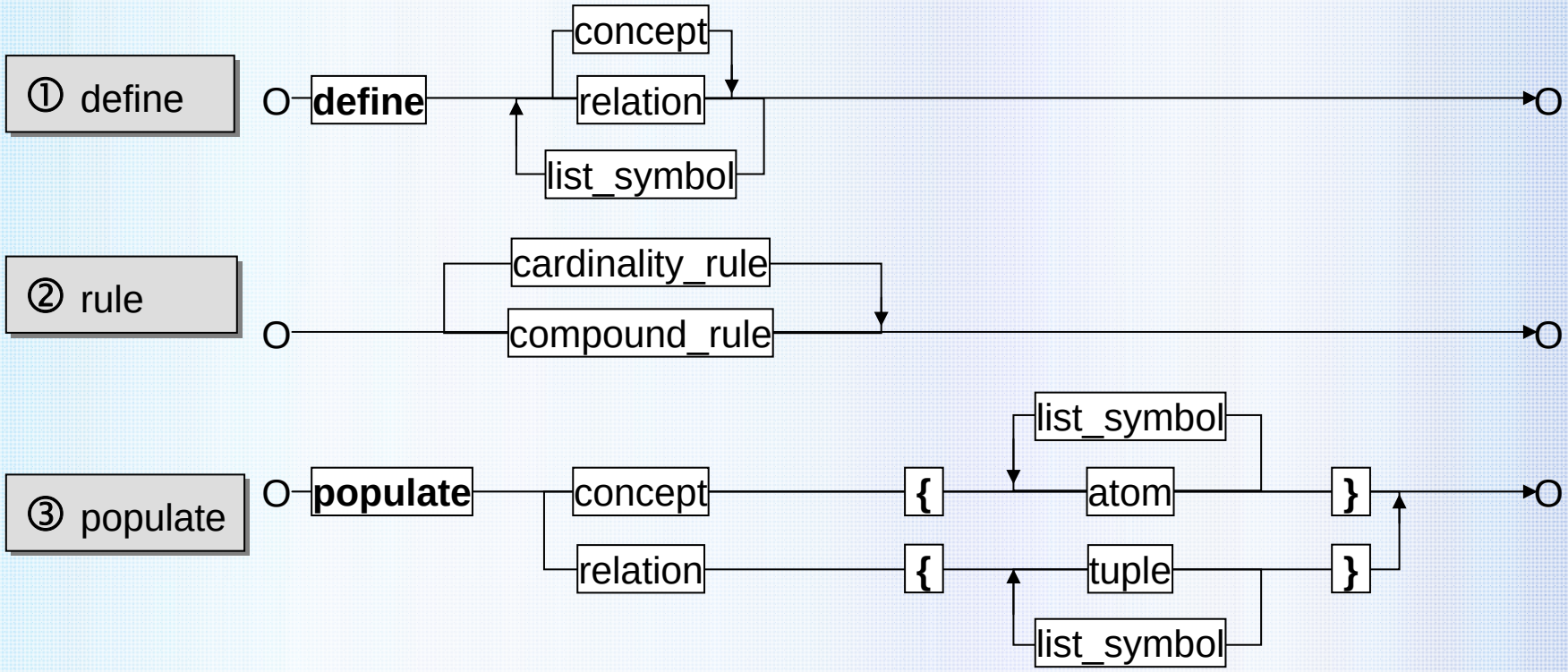


Specification: step 2

detail



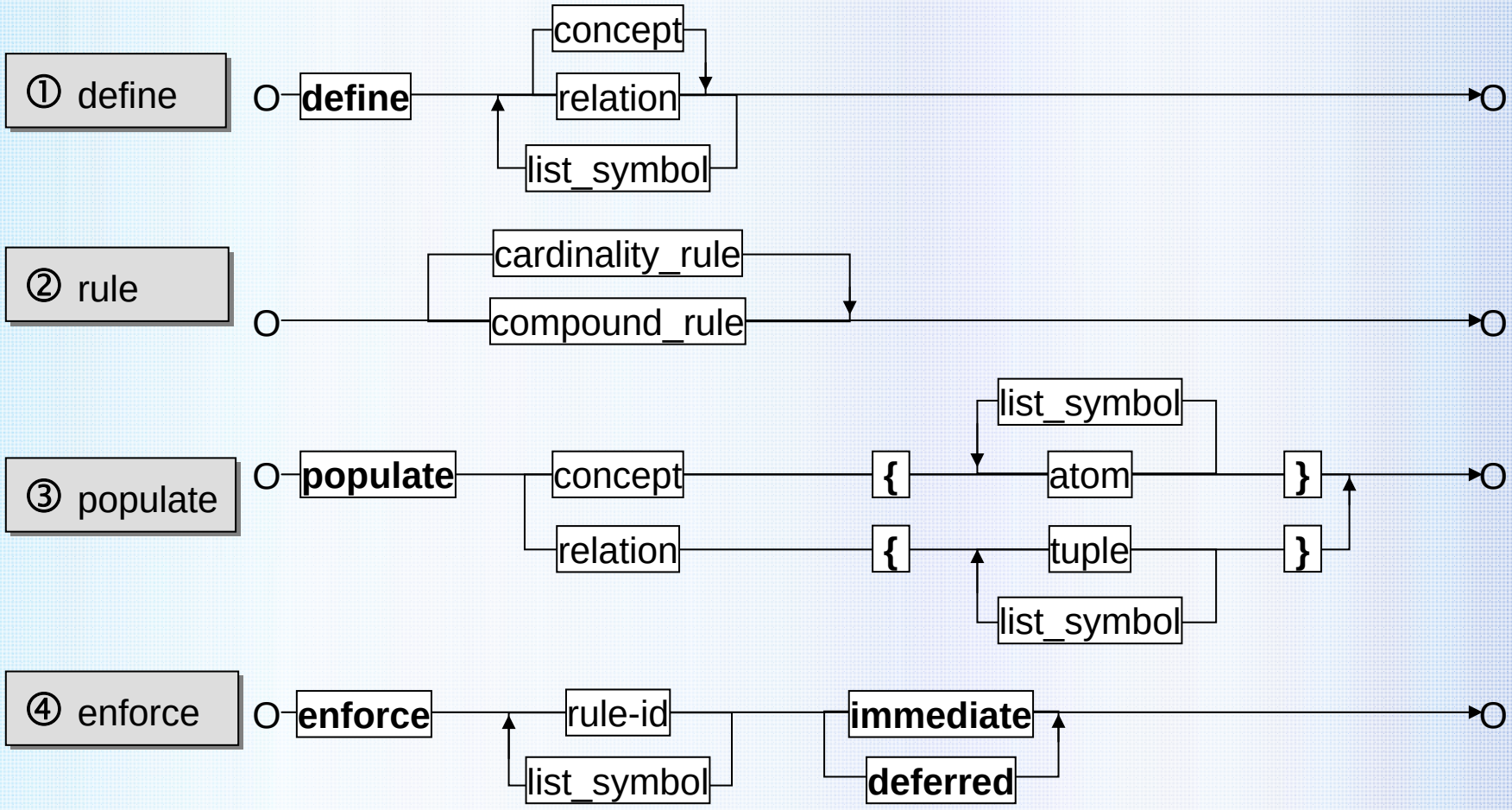
Specification: step 3



Entity integrity, Referential Integrity

Rule violations emerge

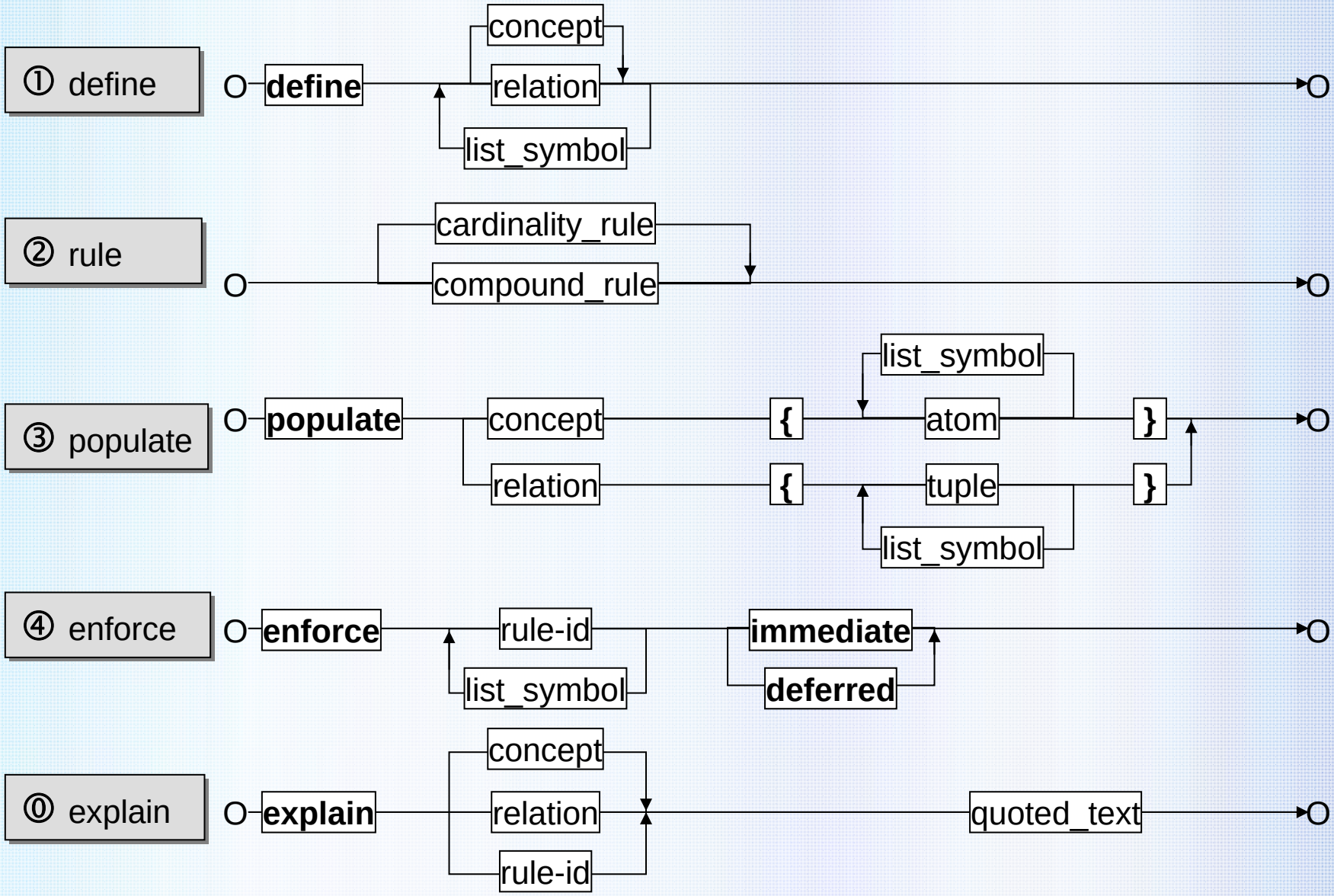
Specification: step 4



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 question *has got an* acceptable-answer

THEN the speaker *thanks you for* your attention